無料電子ブック

学習

MySQL

Free unaffiliated eBook created from
Stack Overflow contributors.

#mysql

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: mysql

It is an unofficial and free MySQL ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official MySQL.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: MySQLをいめる

MySQLは、Oracle Corporationによっておよびサポートされているオープンソースリレーショナルデータベースシステムです。RDBMS

MySQLは、Linuxの、OS X、Windowsなど、のプラットフォームでサポートされています。また、C、C ++、Java、Lua、.Net、Perl、PHP、Python、Rubyなど、ののAPIもえています 。

MariaDBはMySQLのフォークで、 セットがしなります 。これは、ほとんどのアプリケーションでMySQLとにがあります。

バージョン

バージョン	
1.0	1995-05-23
3.19	1996-12-01
3.20	1997-01-01
3.21	1998-10-01
3.22	1999-10-01
3.23	2001122
4.0	2003-03-01
4.1	2004101
5.0	2005-10-01
5.1	20081127
5.5	2010-11-01
5.6	2013-02-01
5.7	2015-10-01

Examples

MySQLでのデータベースの

```
CREATE DATABASE mydb;
```

り

クエリOK、1に0.05

されたデータベース `mydb`

```
USE mydb;
```

り

データベースがされました

MySQLでのテーブルの

```
CREATE TABLE mytable
(
  id              int unsigned NOT NULL auto_increment,
  username        varchar(100) NOT NULL,
  email           varchar(100) NOT NULL,
  PRIMARY KEY     (id)
);
```

`CREATE TABLE mytable`は、 `CREATE TABLE mytable`というしいテーブルをし`mytable`。

`id int unsigned NOT NULL auto_increment`は`id`カラムをします。このタイプのフィールドは、テーブルのレコードにのIDをりてますこの、2つのがじ`id`をつことはできませんレコードの`id`フィールド1からまるにの。

り

クエリOK、をけた00.10

MySQLテーブルにをする

```
INSERT INTO mytable ( username, email )
VALUES ( "myuser", "myuser@example.com" );
```

りの

クエリOK、1に0.06

`varchar` aka `strings`は、をしてすることもできます。

```
INSERT INTO mytable ( username, email )
VALUES ( 'username', 'username@example.com' );
```

を**MySQL**テーブルにする

```
UPDATE mytable SET username="myuser" WHERE id=8
```

りの

    クエリOK、1に0.06

`int`はなしでクエリにできます。とはでむがあります'または" 。

を**MySQL**テーブルにする

```
DELETE FROM mytable WHERE id=8
```

りの

    クエリOK、1に0.06

`id`が8のがされます。

MySQLのについてをする

```
SELECT * FROM mytable WHERE username = "myuser";
```

り

```
+----+----------+---------------------+
| id | username | email               |
+----+----------+---------------------+
|  1 | myuser   | myuser@example.com  |
+----+----------+---------------------+
```

    1セット0.00

のデータベースのリストをする

```
SHOW databases;
```

り

```
+--------------------+
| Databases          |
+--------------------+
| information_schema|
| mydb               |
+--------------------+
```

2セット0.00

"information_schema"はデータベースメタデータへのアクセスをする "マスターデータベース"とえることができます。

のデータベースにテーブルをする

```
SHOW tables;
```

り

```
+----------------+
| Tables_in_mydb |
+----------------+
| mytable        |
+----------------+
```

1セット0.00

テーブルのすべてのフィールドをする

```
DESCRIBE databaseName.tableName;
```

すでにデータベースをしているは、

```
DESCRIBE tableName;
```

り

```
+-----------+-----------------+--------+---------+--------------------+-------+
| Field     | Type            | Null   | Key     | Default            | Extra |
+-----------+-----------------+--------+---------+--------------------+-------+
| fieldname | fieldvaluetype | NO/YES | keytype | defaultfieldvalue |       |
+-----------+-----------------+--------+---------+--------------------+-------+
```

`Extra`は`auto_increment`などがまれます。

`Key`とは、フィールドにするキーのタイプをします。プライマリPRI、ユニークUNI...

nセット0.00

ここで、nはテーブルのフィールドのです。

ユーザーの

まず、ユーザーをし、のデータベース/テーブルにしてユーザーにアクセスをえるがあります。ユーザーのに、このユーザーがどこからできるかをするもあります。

```
CREATE USER 'user'@'localhost' IDENTIFIED BY 'some_password';
```

データベースがホストされているローカルマシンでのみできるユーザーをします。

```
CREATE USER 'user'@'%' IDENTIFIED BY 'some_password';
```

どこからでもできるユーザーをしますローカルマシンをく。

りの

クエリOK、をけた00.00

の

されたデータベースのすべてのテーブルについて、のをユーザーにします。

```
GRANT SELECT, INSERT, UPDATE ON databaseName.* TO 'userName'@'localhost';
```

すべてのデータベースのすべてのテーブルにするすべてのをユーザにしますこれにする。

```
GRANT ALL ON *.* TO 'userName'@'localhost' WITH GRANT OPTION;
```

にしたように、 `*.*`はすべてのデータベースとテーブルをとしていますが、 `databaseName.*`はのデータベースのすべてのテーブルをとしています。 databaseName.tableNameのようにデータベースとテーブルをすることもでき`databaseName.tableName`。

`WITH GRANT OPTION`は、ユーザーがのユーザーにをえるがないは`WITH GRANT OPTION`ください。

は、

```
ALL
```

またはコンマでられたのみわせなリストをすることができます。

```
SELECT
INSERT
UPDATE
DELETE
CREATE
DROP
```

に、スペースまたはSQLでをするまたはのをけるようにしてください。えば、それはのようなをけるのがベストです`table`または`first name`。

そのようなをするがあるは、それらをバックティックの``デリミタ``にいてください。えば

```
CREATE TABLE `table`
(
   `first name` VARCHAR(30)
);
```

こののバックティックりをむクエリはのようになります。

```
 SELECT `first name` FROM `table` WHERE `first name` LIKE 'a%';
```

スキーマの

プロセスリスト

これは、すべてのアクティブなスリープのクエリをそのでし、にどれくらいのだけします。

```
SELECT * FROM information_schema.PROCESSLIST ORDER BY INFO DESC, TIME DESC;
```

これはデフォルトでであるため、にするもうしです。

```
SELECT ID, USER, HOST, DB, COMMAND,
TIME as time_seconds,
ROUND(TIME / 60, 2) as time_minutes,
ROUND(TIME / 60 / 60, 2) as time_hours,
STATE, INFO
FROM information_schema.PROCESSLIST ORDER BY INFO DESC, TIME DESC;
```

ストアドプロシージャの

やワイルドカードのすべての`Stored Procedures`をにできます。

```
SELECT * FROM information_schema.ROUTINES WHERE ROUTINE_DEFINITION LIKE '%word%';
```

オンラインでMySQLをいめるをむ https://riptutorial.com/ja/mysql/topic/302/mysqlをいめる

2: 1

き

1のえ1Mは、の、に、あるののがののののにするにします。

1Mはです。つまり、1Mのにクエリをするときはいつでも、 '1'をってのテーブルの 'many'をできますが、 1つの「1」をします。

ほとんどの、1Mのでするには、 キーとキーをするがあります。

キーは、そののののがのエンティティをすのです。または、キーのをすると、に1つのになります。のをすると、EMP_IDは1のをします。のEMP_IDをすると、するをすのがされます。

キーは、のテーブルのキーにするテーブルのです。のから、EMPLOYEESのMGR_IDはキーです。に2つのテーブルをするには、あるテーブルのキーとのテーブルのキーにづいてします。

Examples

の

マネージャであるすべてのが1のをし、すべてのが1のしかたないをえてみましょう。

これにより、2つのテーブルがされます。

EMP_ID	ファーストネーム		MGR_ID
E01	ジョニー	アップルシード	M02
E02	エリン	マックモア	M01
E03	コルビー		M03
E04	ロン	ソンスワン	M01

マネージャー

MGR_ID	ファーストネーム	
M01	で	マックイーン
M02	ボッシー	パンツ
M03	バレル	ジョーンズ

のマネ―ジャによってされるをする

```
SELECT e.emp_id , e.first_name , e.last_name FROM employees e INNER JOIN managers m ON m.mgr_id
= e.mgr_id WHERE m.mgr_id = 'M01' ;
```

EMP_ID	ファ―ストネ―ム	
E02	エリン	マックモア
E04	ロン	ソンスワン

に、たちがするマネ―ジャ―ごとに、1のがってきます。

1ののマネ―ジャをする

このをるときは、ののをしてください。

```
SELECT m.mgr_id , m.first_name , m.last_name FROM managers m INNER JOIN employees e ON e.mgr_id
= m.mgr_id WHERE e.emp_id = 'E03' ;
```

MGR_ID	ファ―ストネ―ム	
M03	バレル	ジョ―ンズ

これはののですから、たちがいわせるすべてのにして、するマネ―ジャ―は1つしかされません。

オンラインで1をむ https://riptutorial.com/ja/mysql/topic/9600/1

3: Docker-ComposeでMysqlコンテナをインストールする

Examples

ドッカーのによるな

これは、dockerをしてmysqlサーバをするなです

1.-する**docker-compose.yml**

すべてのプロジェクトでじコンテナをするは、HOME_PATHにPATHをするがあります。プロジェクトごとにするは、プロジェクトにドッカーディレクトリをすることができます。

```
version: '2'
services:
  cabin_db:
    image: mysql:latest
    volumes:
      - "./.mysql-data/db:/var/lib/mysql"
    restart: always
    ports:
      - 3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: rootpw
      MYSQL_DATABASE: cabin
      MYSQL_USER: cabin
      MYSQL_PASSWORD: cabinpw
```

2.それをする

```
cd PATH_TO_DOCKER-COMPOSE.YML
docker-compose up -d
```

3.-サーバーにする

```
mysql -h 127.0.0.1 -u root -P 3306 -p rootpw
```

ハレイ

4.-サーバー

```
docker-compose stop
```

オンラインでDocker-ComposeでMysqlコンテナをインストールするをむ
https://riptutorial.com/ja/mysql/topic/4458/docker-composeでmysqlコンテナをインストールする

4: ENUM

Examples

なぜENUM

ENUMはのをするをします。のオプションをつがにします。

```
reply ENUM('yes', 'no')
gender ENUM('male', 'female', 'other', 'decline-to-state')
```

はです。

```
INSERT ... VALUES ('yes', 'female')
SELECT ... --> yes female
```

としてのTINYINT

たちがっているとしましょう

```
type ENUM('fish','mammal','bird')
```

わりに

```
type TINYINT UNSIGNED
```

プラス

```
CREATE TABLE AnimalTypes (
    type TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL  COMMENT "('fish','mammal','bird')",
    PRIMARY KEY(type),
    INDEX(name)
) ENGINE=InnoDB
```

これはのテーブルににによくています。

、そしてENUMよりいかいか

- するINSERT `type`をするがある
- いSELECTをするためにJOINするがありますENUMはあなたにのストリングをえます
- よりいしいタイプのにこのテーブルにします。 ENUMでは、ALTER TABLEをするがあります。
- じいずれのテクニック255までのでも1バイトしかかかりません。
- データののもあります `TINYINT`はなを`TINYINT`ます。 `ENUM`はなストリングにしますなSQLモー

ドがになっていないはされます。ルックアップテーブルにキーをすることで、 `TINYINT`データを`TINYINT`させることができます。これは、なクエリ/をしても、のテーブルにするためのコストはわずかです。 `FOREIGN KEYs`はではありません

わりの**VARCHAR**

たちがっているとしましょう

```
type ENUM('fish','mammal','bird')
```

わりに

```
type VARCHAR(20)  COMENT "fish, bird, etc"
```

これは、しいタイプがにされるというで、くである。

、そしてENUMよりいかいか

- じINSERTにをする
- しますかINSERTではタイプミスがかれません
- じSELECTのがされます
- するよりくのスペースがされる

しいオプションをする

```
ALTER TABLE tbl MODIFY COLUMN type ENUM('fish','mammal','bird','insect');
```

ノート

- MODIFY COLUMNのすべてのケースとに、 `NOT NULL`と々していたのもめなければなり`NOT NULL`。それのはわれます。
- あなたがリストのにし、リストが256ののにあるは 、 `ALTER`にスキーマをすることでわれます。つまり、いテーブルコピーはありません。 バージョンのMySQLにはこのがありませんでした。

NULLと**NOT NULL**

NULLと 'bad-value'がNULLなとNULLでないにされているときにどうなるかの。また、 `+0`をしてへのキャストのもされます。

```
CREATE TABLE enum (
    e     ENUM('yes', 'no')   NOT NULL,
    enull ENUM('x', 'y', 'z')     NULL
        );
INSERT INTO enum (e, enull)
    VALUES
```

```
        ('yes', 'x'),
        ('no',  'y'),
        (NULL,  NULL),
        ('bad-value', 'bad-value');
Query OK, 4 rows affected, 3 warnings (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 3

mysql>SHOW WARNINGS;
+---------+------+-------------------------------------------+
| Level   | Code | Message                                   |
+---------+------+-------------------------------------------+
| Warning | 1048 | Column 'e' cannot be null                 |
| Warning | 1265 | Data truncated for column 'e' at row 4    |
| Warning | 1265 | Data truncated for column 'enull' at row 4 |
+---------+------+-------------------------------------------+
3 rows in set (0.00 sec)
```

それらののののテーブルにはがっていますかこれは "+0"をしてにキャストし、がされているかをします。

```
mysql>SELECT e, e+0 FROM enum;
+-----+-----+
| e   | e+0 |
+-----+-----+
| yes |   1 |
| no  |   2 |
|     |   0 |  -- NULL
|     |   0 |  -- 'bad-value'
+-----+-----+
4 rows in set (0.00 sec)

mysql>SELECT enull, enull+0 FROM enum;
+-------+---------+
| enull | enull+0 |
+-------+---------+
| x     |       1 |
| y     |       2 |
| NULL  |    NULL |
|       |       0 |  -- 'bad-value'
+-------+---------+
4 rows in set (0.00 sec)
```

オンラインでENUMをむ https://riptutorial.com/ja/mysql/topic/4425/enum

5: GRANTによるセキュリティ

Examples

ベストプラクティス

ルートおよびのSUPERユーザーを

```
GRANT ... TO root@localhost ...
```

これにより、のサーバーからのアクセスがされます。あなたはにのにSUPERをすべきです、そして、らはらのをするべきです。アプリケーションはSUPERをつべきではありません。

アプリケーションログインは、する1つのデータベースにします。

```
GRANT ... ON dbname.* ...
```

そうすれば、アプリケーションコードをハックするはdbnameをえることができません。これは、のいずれかによってさらにされます。

```
GRANT SELECT ON dname.* ...    -- "read only"
GRANT ... ON dname.tblname ... -- "just one table"
```

みみには、「な」ものがなもあります

```
GRANT SELECT, CREATE TEMPORARY TABLE ON dname.* ...    -- "read only"
```

あなたがうように、なセキュリティはありません。のここでのポイントは、あなたがハッカーをらせるためにいくつかのことをうことができることです。 なたちのためにじことがこります。

ごくまれに、rootだけがなかをうためにアプリケーションがながあります。これは、 `SECURITY DEFINER`をつ「ストアドプロシージャ」およびルートがそれをするをしてできます。それは、SPがうことだけをします。たとえば、あるのテーブルでのアクションがされます。

ホスト user @ hostの

「ホスト」は、ホストまたはIPアドレスのいずれかです。また、ワイルドカードもできます。

```
GRANT SELECT ON db.* TO sam@'my.domain.com' IDENTIFIED BY 'foo';
```

これらはするがあります

```
localhost -- the same machine as mysqld
'my.domain.com' -- a specific domain; this involves a lookup
```

```
'11.22.33.44' -- a specific IP address
'192.168.1.%' -- wild card for trailing part of IP address.  (192.168.% and 10.% and 11.% are
"internal" ip addresses.)
```

`localhost`をすると、サーバーのセキュリティにします。ベストプラクティスのためには、 `root`はlocalhostをしてのみされるべきです。によっては、これらはじことをします `0.0.0.1`と`::1`。

オンラインでGRANTによるセキュリティをむ https://riptutorial.com/ja/mysql/topic/5131/grantによるセキュリティ

6: JOINSidとじのテーブルを3つします。

Examples

じのに3つのテーブルをする

```
CREATE TABLE Table1 (
    id INT UNSIGNED NOT NULL,
    created_on DATE NOT NULL,
    PRIMARY KEY (id)
)
CREATE TABLE Table2 (
    id INT UNSIGNED NOT NULL,
    personName VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
)
CREATE TABLE Table3 (
    id INT UNSIGNED NOT NULL,
    accountName VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
)
```

テーブルをした、クエリをしてじ3つのテーブルのIDをすることができます

```
SELECT
    t1.id AS table1Id,
    t2.id AS table2Id,
    t3.id AS table3Id
FROM Table1 t1
LEFT JOIN Table2 t2 ON t2.id = t1.id
LEFT JOIN Table3 t3 ON t3.id = t1.id
```

オンラインでJOINSidとじのテーブルを3つします。をむ
https://riptutorial.com/ja/mysql/topic/9921/joins-idとじのテーブルを3つします-

7: JSON

き

MySQL 5.7.8、MySQLはJSONJavaScript Object Notationドキュメントのデ―タににアクセスできるネイティブなJSONデ―タをサポ―トしています。
https://dev.mysql.com/doc/refman/5.7/ja/json.html

MySQL 5.7.8、MySQLにはJSONタイプがしています。くのが、JSONデ―タをログ･タイムのテキストにしていますが、JSONタイプはなります。デ―タはにバイナリでされます。これにより、みみにテキストをするオ―バ―ヘッドがされます。

Examples

キ―とJSONフィ―ルドをつシンプルなテ―ブルをする

```
CREATE TABLE table_name (
    id INT NOT NULL AUTO_INCREMENT,
    json_col JSON,
    PRIMARY KEY(id)
);
```

なJSONをする

```
INSERT INTO
    table_name (json_col)
VALUES
    ('{"City": "Galle", "Description": "Best damn city in the world"}');
```

これはですが、JSONのキ―をでむがあるため、をでむがあります。クエリがすると、デ―タはバイナリでされます。

デ―タをJSONフィ―ルドにします。

これは、メンバの1つがのでされたテ―ブルへののであるjsonをします。

```
INSERT INTO myjson(dict)
VALUES('{"opening":"Sicilian","variations":["pelikan","dragon","najdorf"]}');
```

もう、とのにするがあることにしてください。をでむがあります。

JSONフィ―ルドの

のでは、デ―タをJSONフィ―ルドにするをてきました。もしそのフィ―ルドをしたいのであれ

ばのでは、というの`variations`に*scheveningen*をします。

```
UPDATE
    myjson
SET
    dict=JSON_ARRAY_APPEND(dict,'$.variations','scheveningen')
WHERE
    id = 2;
```

ノート

1. jsonの`$.variations`。 $はjsonのドキュメントをします。 mysqlによってされるjsonパスのなについては、 https: //dev.mysql.com/doc/refman/5.7/en/json-path-syntax.htmlをしてください。
2. jsonフィールドをしてクエリをするはまだないので、このではキーをしています。

`SELECT * FROM myjson`をすると

```
+----+-------------------------------------------------------------------------------------
-+
| id | dict
|
+---+--------------------------------------------------------------------------------------
+
| 2  | {"opening": "Sicilian", "variations": ["pelikan", "dragon", "najdorf", "scheveningen"]}
|
+----+-------------------------------------------------------------------------------------
-+
1 row in set (0.00 sec)
```

データをJSONタイプにする

これは、なjsonをMySQL JSONにします。

```
 SELECT CAST('[1,2,3]' as JSON) ;
 SELECT CAST('{"opening":"Sicilian","variations":["pelikan","dragon","najdorf"]}' as JSON);
```

Jsonオブジェクトとをする

`JSON_OBJECT`はJSONオブジェクトをします

```
 SELECT JSON_OBJECT('key1',col1 , 'key2',col2 , 'key3','col3') as myobj;
```

`JSON_ARRAY`はJSONもします

```
 SELECT JSON_ARRAY(col1,col2,'col3') as myarray;
```

myobj.key3とmyarray [2]はとして "col3"です。

JSONデータもしています

```
 SELECT JSON_OBJECT("opening","Sicilian",
"variations",JSON_ARRAY("pelikan","dragon","najdorf") ) as mymixed ;
```

オンラインでJSONをむ https://riptutorial.com/ja/mysql/topic/2985/json

8: JSONからをする

き

MySQL 5.7.8は、ネイティブJSONタイプをサポートしています。 jsonオブジェクトをするさまざまながありますが、さまざまなでメンバーにアクセスしたりむことができます。

なは`JSON_EXTRACT` 、 `->`および`->>`はよりフレンドリーです。

- JSON_EXTRACTjson_doc、path [、...]
- JSON_EXTRACTjson_doc、path
- JSON_EXTRACTjson_doc、path1、path2

パラ メーター

パラ メータ	
json_doc	なJSON
パス	メンバーパス

MySQL 5.7リファレンスマニュアルでされている

- パスによるのした

  それらのがのをすがある、したは、それらをしたパスにするでとしてオートラップされます。それの、りはのしたです。

- `NULL`
  - argemuntはNULLです
  - しないパス

  いずれかのがNULLであるか、パスがドキュメントのをつけることができないは、NULLをします。

Examples

JSONのをみむ

@myjsonをJSONとしてするきをむ

```
SET @myjson = CAST('["A","B",{"id":1,"label":"C"}]' as JSON) ;
```

メンバーを`SELECT`ください

```
SELECT
  JSON_EXTRACT( @myjson , '$[1]' ) ,
  JSON_EXTRACT( @myjson , '$[*].label') ,
  JSON_EXTRACT( @myjson , '$[1].*' ) ,
  JSON_EXTRACT( @myjson , '$[2].*')
;
-- result values:
'\"B\"', '[\"C\"]', NULL, '[1, \"C\"]'
-- visually:
"B", ["C"], NULL, [1, "C"]
```

JSON

`path`によって`->`は`->>`、つつ`->>`でまれていないです。

```
SELECT
  myjson_col->>'$[1]' , myjson_col->'$[1]' ,
  myjson_col->>'$[*].label' ,
  myjson_col->>'$[1].*' ,
  myjson_col->>'$[2].*'
FROM tablename ;
 -- visuall:
    B, "B" , ["C"], NULL, [1, "C"]
 --^^^ ^^^
```

だから`col->>path`は`JSON_UNQUOTE(JSON_EXTRACT(col,path))`としくなり
`JSON_UNQUOTE(JSON_EXTRACT(col,path))`

- `->`とに、のにすように、 - `>>`はにEXPLAINのにされます。

```
mysql> EXPLAIN SELECT c->>'$.name' AS name
    ->     FROM jemp WHERE g > 2\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: jemp
   partitions: NULL
         type: range
possible_keys: i
          key: i
      key_len: 5
          ref: NULL
         rows: 2
     filtered: 100.00
        Extra: Using where
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Note
   Code: 1003
Message: /* select#1 */ select
json_unquote(json_extract(`jtest`.`jemp`.`c`,'$.name')) AS `name` from
`jtest`.`jemp` where (`jtest`.`jemp`.`g` > 2)
```

```
1 row in set (0.00 sec)
```

インラインパス+についてむ

オンラインでJSONからをするをむ https://riptutorial.com/ja/mysql/topic/9042/jsonからをする

9: LOAD DATA INFILE

1. LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
2. INTO TABLE tbl_name
3. [CHARACTER SET charset]
4. [{FIELDS | COLUMNS} ['string'によってしました] [[OPTIONALLY] 'ENCLOSED BY' char ']]
5. [LINES [STARTING BY 'string'] [TERMINATED BY 'string']]
6. [する{| ROWS}]
7. [col_name_or_user_var、...]
8. [SET col_name = expr、...]

Examples

LOAD DATA INFILEをしてのデータをデータベースにロードする

データベースにロードするCSVが ';'でられているとして、のをえてみましょう。

```
1;max;male;manager;12-7-1985
2;jack;male;executive;21-8-1990
.
.
.
1000000;marta;female;accountant;15-6-1992
```

するテーブルをします。

```
CREATE TABLE `employee` ( `id` INT NOT NULL ,
                          `name` VARCHAR NOT NULL,
                          `sex` VARCHAR NOT NULL ,
                          `designation` VARCHAR NOT NULL ,
                          `dob` VARCHAR NOT NULL   );
```

のクエリをして、そのテーブルにをします。

```
LOAD DATA INFILE 'path of the file/file_name.txt'
INTO TABLE employee
FIELDS TERMINATED BY ';' //specify the delimiter separating the values
LINES TERMINATED BY '\r\n'
(id,name,sex,designation,dob)
```

がのをえてみましょう。

```
1;max;male;manager;17-Jan-1985
2;jack;male;executive;01-Feb-1992
.
.
.
1000000;marta;female;accountant;25-Apr-1993
```

このようにするに、 dobカラムのフォーマットをすることができます。

```
LOAD DATA INFILE 'path of the file/file_name.txt'
INTO TABLE employee
FIELDS TERMINATED BY ';' //specify the delimiter separating the values
LINES TERMINATED BY '\r\n'
(id,name,sex,designation,@dob)
SET date = STR_TO_DATE(@date, '%d-%b-%Y');
```

LOAD DATA INFILEのこのでは、なのすべてをしていません。

ここでLOAD DATA INFILEのをることができます 。

CSVファイルをMySQLテーブルにインポートする

のコマンドは、CSVファイルを、じをつMySQLテーブルにインポートして、CSVおよびエスケープルールをします。

```
load data infile '/tmp/file.csv'
into table my_table
fields terminated by ','
optionally enclosed by '"'
escaped by '"'
lines terminated by '\n'
ignore 1 lines; -- skip the header row
```

データをしてロードする

LOAD DATA INFILEコマンドをしてのデータをにりむと、によってインポートがすることがよくあります。このをするはいくつかあります。

LOAD DATA LOCAL

このオプションがサーバーでになっているは、サーバーではなくクライアントコンピューターにするファイルをみむためにできます。は、ののがされることです。

```
LOAD DATA LOCAL INFILE 'path of the file/file_name.txt'
INTO TABLE employee
```

LOAD DATA INFILE 'fname' REPLACE

replaceキーワードをすると、のキーまたはキーがすると、のがしいものにきえられます

```
LOAD DATA INFILE 'path of the file/file_name.txt'
REPLACE INTO TABLE employee
```

LOAD DATA INFILE 'fname' IGNORE

REPLACEとはに、のはされ、しいはされます。このは、のLOCALとです。ただし、ファイルはクライアントコンピュータにするはありません。

```
LOAD DATA INFILE 'path of the file/file_name.txt'
IGNORE INTO TABLE employee
```

テーブルをしてロード

によっては、すべてのをまたはすることはなではないかもしれません。ののにづいてするがあるかもしれません。その、のは、テーブルにロードしてそこからすることです。

```
INSERT INTO employee SELECT * FROM intermediary WHERE ...
```

インポート・エクスポート

インポート

```
SELECT a,b,c  INTO OUTFILE 'result.txt' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n' FROM table;
```

する

```
LOAD DATA INFILE 'result.txt' INTO TABLE table;
```

オンラインでLOAD DATA INFILEをむ https://riptutorial.com/ja/mysql/topic/2356/load-data-infile

10: MyISAMエンジン

にわたり、InnoDBは、なくともサポートされているバージョンであるMyISAMよりもほぼにされています。するに、になテーブルのをいて、MyISAMはしないでください。

InnoDBにべてMyISAMののの1つは、ディスクになスペースが23さいことです。

InnoDBがめてしたとき、MyISAMはとしてなエンジンでした。しかし、XtraDBと5.6がしたことで、InnoDBはほとんどのベンチマークでMyISAMよりも「い」になりました。

のメジャーバージョンでは、になInnoDBテーブルをし、システムテーブルをInnoDBにすることで、MyISAMのがなくなるというがあります。

Examples

ENGINE = MyISAM

```
CREATE TABLE foo (
     ...
) ENGINE=MyISAM;
```

オンラインでMyISAMエンジンをむ https://riptutorial.com/ja/mysql/topic/4710/myisamエンジン

11: MyISAMからInnoDBへの

Examples

な

```
ALTER TABLE foo ENGINE=InnoDB;
```

これはテーブルをしますが、エンジンのいはしません。ほとんどのいは、にさなテーブルではになりません。しかし、したテーブルのは、のもするがあります。 にする

1つのデータベースですべてのテーブルをする

1つのデータベースのすべてのテーブルをにするには、をします。

```
SET @DB_NAME = DATABASE();

SELECT  CONCAT('ALTER TABLE `', table_name, '` ENGINE=InnoDB;') AS sql_statements
FROM    information_schema.tables
WHERE   table_schema = @DB_NAME
AND     `ENGINE` = 'MyISAM'
AND     `TABLE_TYPE` = 'BASE TABLE';
```

> `DATABASE()`がするには、データベースにしているがあり`NULL`。それのは、`NULL`がされ`NULL`。これは、データベースをせずにできるようにするため、サーバにののmysqlクライアントにされます。

このSQLをして、データベースのすべての`MyISAM`テーブルをします。

に、をコピーしてSQLクエリをします。

オンラインでMyISAMからInnoDBへのをむ https://riptutorial.com/ja/mysql/topic/3135/myisamからinnodbへの

12: MySQL 5.7+のデフォルトのrootパスワードをしてリセットする

き

MySQL 5.7、MySQLをインストールするときにrootアカウントをするはなく、ルートパスワードをえるもありません。デフォルトでは、サーバをすると、デフォルトのパスワードが`mysqld.log`ファイルにされます。そのパスワードをしてシステムにログインするがあり、パスワードをするがあります。

このをしてデフォルトのルートパスワードをしてリセットすることは、MySQL 5.7

Examples

サーバーのにがこるか

サーバのデータディレクトリがの

- サーバーがされます。
- SSLとキーファイルがデータディレクトリにされます。
- validate_passwordプラグインがインストールされ、になっています。
- スーパーユーザーアカウント 'root' @ 'localhost'がされます。スーパーユーザーのパスワードがされ、エラーログファイルにされます。

のパスワードをしてルートパスワードをする

デフォルトの「root」パスワードをするには

```
shell> sudo grep 'temporary password' /var/log/mysqld.log
```

されたパスワードでログインし、スーパーユーザーアカウントのカスタムパスワードをして、できるだけくrootパスワードをします。

```
shell> mysql -uroot -p

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass5!';
```

MySQLのvalidate_passwordプラグインはデフォルトでインストールされます。これには、パスワードになくとも1つの、1つの、1つの、および1つのがまれ、パスワードのさのが8であるがあります。

UNIXソケットファイルの "/ var / run / mysqld"がしないにrootパスワードをリセ

ットする "

がパスワードをれたら、はエラーにうでしょう。

```
$ mysql -u root -p
```

パスワードをする

ERROR 104528000ユーザー 'root' @ 'localhost'のアクセスがされましたパスワードは「はい」です

はにステータスをってをしようとしました

```
$ systemctl status mysql.service
```

mysql.service - ロードされたMySQLコミュニティサーバ/lib/systemd/system/mysql.service; enabled;ベンダープリセットenアクティブアクティブなThu 2017-06-08 14:31:33 IST; 38s

それから、コード`mysqld_safe --skip-grant-tables &`をしましたが、エラーがます

    mysqld_safe UNIXソケットファイルのディレクトリ '/ var / run / mysqld'がしません。

```
$ systemctl stop  mysql.service
$ ps -eaf|grep mysql
$ mysqld_safe --skip-grant-tables &
```

    はした

```
$ mkdir -p /var/run/mysqld
$ chown mysql:mysql /var/run/mysqld
```

は`mysqld_safe --skip-grant-tables &`とじコードをい、get

    mysqld_safe / var / lib / mysqlのデータベースをってmysqldデーモンをする

が`$ mysql -u root`をうと、はのようになるでしょう

サーバーのバージョン5.7.18-0ubuntu0.16.04.1Ubuntu

Copyrightc2000、2017、Oracleおよび/またはその。

Oracleは、Oracle Corporationおよびそののです。そののは、それぞれののです。

タイプ 'help;'ヘルプは '\ h'をしてください。のステートメントをクリアするには '\ c'とします。

mysql>

    すぐパスワードをする

```
mysql> use mysql
mysql> describe user;
```

テーブルとカラムののためにテーブルをむこのをオフにすると、-A

データベースの

```
mysql> FLUSH PRIVILEGES;
mysql> SET PASSWORD FOR root@'localhost' = PASSWORD('newpwd');
```

またはどこからでもできるmysql rootアカウントをおちのは、のようにするがあります。

```
UPDATE mysql.user SET Password=PASSWORD('newpwd') WHERE User='root';
```

```
   USE mysql
   UPDATE user SET Password = PASSWORD('newpwd')
   WHERE Host = 'localhost' AND User = 'root';
```

どこからでもアクセスできるrootアカウントをおちの

```
 USE mysql
 UPDATE user SET Password = PASSWORD('newpwd')
 WHERE Host = '%' AND User = 'root';`enter code here
```

はmysqlを`quit`してstop / startするがあります

```
FLUSH PRIVILEGES;
sudo /etc/init.d/mysql stop
sudo /etc/init.d/mysql start
```

もう`mysql -u root -p 'をし、しいパスワードをって

    mysql>

オンラインでMySQL 5.7+のデフォルトのrootパスワードをしてリセットするをむ
https://riptutorial.com/ja/mysql/topic/9563/mysql-5-7plusのデフォルトのrootパスワードをしてリセットする

13: mysqldumpをったバックアップ

- mysqldump -u [ユーザ] -p [パスワード] [そののオプション] db_name> dumpFileName.sql /// のデータベースをバックアップするには
- mysqldump -u [ユーザ] -p [パスワード] [そののオプション] db_name [tbl_name1 tbl_name2 tbl_name2 ...]> dumpFileName.sql /// 1つのテーブルをバックアップする
- mysqldump -u [ユーザ] -p [パスワード] [そののオプション] --databases db_name1 db_name2 db_name3 ...> dumpFileName.sql /// 1つのなデータベースをバックアップする
- mysqldump -u [ユーザ] -p [パスワード] [そののオプション] --all-databases> dumpFileName.sql /// MySQLサーバをバックアップする

パラメータ

オプション	
-	サーバログインオプション
`-h` -- `--host`	のホストIPアドレスまたはホスト。デフォルトは`localhost 127.0.0.1`です。`-h localhost`
`-u` -- `--user`	MySQLユーザー
`-p` -- `--password`	MySQLパスワード。 `-p`をする、オプションとパスワードのにスペースをれてはいけません。 `-pMyPassword`
-	ダンプオプション
`--add-drop-database`	`CREATE DATABASE`のに`DROP DATABASE`をします。 サーバのデータベースをするにです。
`--add-drop-table`	`CREATE TABLE`ステートメントのに`DROP TABLE`ステートメントをしてください。 サーバのテーブルをきえるにです。
`--no-create-db`	ダンプの`CREATE DATABASE`をにします。これは、ダンプしているデータベースが、ダンプをロードするサーバにすでにしていることをしているにです。
`-t` -- `--no-create-info`	ダンプのすべての`CREATE TABLE`ステートメントをします。これは、テーブルのデータだけをダンプし、ダンプファイルをしてのデータベース/サーバののテーブルにデータをれるにです。
`-d` -- `--no-data`	テーブルをきまないでください。これは、 `CREATE TABLE`ステートメントだけをダンプします。 「テンプレート」データベースのにちます
`-R` - `--routines`	ダンプにストアドプロシージャ/をインクルードします。

オプション	
`-K` -- `--disable-keys`	データをするにテーブルのキーをにし、データをしたにキーをにします。これにより、ではないインデックスをつMyISAMテーブルでのみがされます。

`mysqldump`のは、するためにされたバージョンのMySQLユーティリティとのあるシーケンシャルなSQLをむくコメントされたファイルですのバージョンとのにをっていますが、のはありません。したがって、 `mysqldump`データベースのは、それらののをみます。に、このファイル

- `DROP`のされたテーブルまたはビュー
- そのテーブルまたはビューを`CREATE`
- データでダンプされたテーブル `--no-data`オプションなしでは、
  - テーブルを`LOCK`する
  - 1つでののすべてのを`INSERT`
- `UNLOCK TABLES`
- のすべてのテーブルとビューについてをりします
- `DROP` sがのルーチンがま
- `CREATE`そのルーチン
- のすべてのルーチンでじことをりす

テーブルの`CREATE`に`DROP`するということは、スキーマがする、スキーマがであるかどうかにかかわらず、リストアのために`mysqldump`ファイルをすると、そこにデータがまたはきされることをします。

Examples

データベースまたはテーブルのバックアップをする

データベースのスナップショットをする

```
mysqldump [options] db_name > filename.sql
```

のデータベースのスナップショットをする

```
mysqldump [options] --databases db_name1 db_name2 ... > filename.sql
mysqldump [options] --all-databases > filename.sql
```

1つまたはのテーブルのスナップショットをする

```
mysqldump [options] db_name table_name... > filename.sql
```

1つのテーブルをいたスナップショットをします。

```
mysqldump [options] db_name --ignore-table=tbl1 --ignore-table=tbl2 ... > filename.sql
```

ファイル.sqlはにスタイルのです。のがします。

ユ―ザ―とパスワ―ドの

```
> mysqldump -u username -p [other options]
Enter password:
```

コマンドラインでパスワ―ドをするがあるスクリプトなど、- -pオプションのろにスペ―スをれずにできます。

```
> mysqldump -u username -ppassword [other options]
```

パスワ―ドにスペ―スやがまれているは、シエル/システムにじてエスケ―プをしてください。

オプションで、はのとおりです。

```
> mysqldump --user=username --password=password [other options]
```

コマンドでのパスワ―ドのは、セキュリティのからされていません。

デ―タベ―スまたはテ―ブルのバックアップをする

```
mysql [options] db_name < filename.sql
```

ごください

- db_nameはのデ―タベ―スであるがあります。
- されたユ―ザは、 filename.sqlのすべてのコマンドをするのになをっていfilename.sql。
- ファイル.sqlはにスタイルのです。のがします。
- ダンプするテ―ブルをすることはできますが、ロ―ドするテ―ブルはできません。これはfilename.sqlでうがありfilename.sql。

あるいは、 **MySQL**コマンドライン ツ―ルで、sourceコマンドをしてリストアするまたはのスクリプトをすることができます

```
source filename.sql
```

または

```
\. filename.sql
```

されたリモ―トサ―バからの**mysqldump**

のためにワイヤでをするために、--compressオプションmysqldump。

```
mysqldump -h db.example.com -u username -p --compress dbname > dbname.sql
```

ソース dbをロックしたくないは、-- `--lock-tables=false`もめるがあり`--lock-tables=false`。 しかし、そのようににしたdbイメージをることはできません。

ファイルをしてするには、 `gzip`パイプすることもできます。

```
mysqldump -h db.example.com -u username -p --compress dbname | gzip --stdout > dbname.sql.gz
```

せずに**gzip**された**mysqldump**ファイルをする

```
gunzip -c dbname.sql.gz | mysql dbname -u username -p
```

`-c`はstdoutにをきむことをします。

して**Amazon S3**にバックアップ

なMySqlインストールをにバックアップし、なローカルストレージがないは、Amazon S3バケットにダンプしてすることができます。コマンドのとしてDBパスワードをせずにこれをうのもよいです

```
mysqldump -u root -p --host=localhost --opt --skip-lock-tables --single-transaction \
        --verbose --hex-blob --routines --triggers --all-databases |
    gzip -9 | s3cmd put - s3://s3-bucket/db-server-name.sql.gz
```

パスワードのをめるプロンプトがされ、そのにバックアップがされます。

ある**MySQL**サーバからの**MySQL**サーバへデータをする

あるサーバーからのサーバーにデータベースをコピーするがあるは、の2つのがあります。

オプション**1**

1. ダンプファイルをソースサーバーにする
2. ダンプファイルをコピーのサーバーにコピーする
3. ダンプファイルをターゲットサーバーにロードする

ソースサーバーで

```
mysqldump [options] > dump.sql
```

サーバーで、ダンプ·ファイルをコピーしてのコマンドをします。

```
mysql [options] < dump.sql
```

オプション**2**

サーバーがホストサーバーにできるは、パイプラインをしてデータベースをあるサーバーからのサーバーにコピーできます。

サーバーで

```
mysqldump [options to connect to the source server] | mysql [options]
```

に、スクリプトはサーバーでされ、にプッシュされます。いずれのも、オプション1よりもはるかにであるがい。

ストアドプロシージャとをしたバックアップデータベース

デフォルトではストアドプロシージャとによって、または`mysqldump`によってされないは、パラメータ`--routines` または`-R` をするがあります

```
mysqldump -u username -p -R db_name > dump.sql
```

`--routines`をする`--routines` 、とのタイムスタンプはされず、わりに`mysql.proc`のをダンプして`--routines`するがあり`mysql.proc`。

オンラインでmysqldumpをったバックアップをむ
https://riptutorial.com/ja/mysql/topic/604/mysqldumpをったバックアップ

14: mysqlimport

パラメーター

パラメータ	
`--delete -D`	テキストファイルをインポートするにテーブルをにする
`--fields-optionally-enclosed-by`	フィールドをでむをする
`--fields-terminated-by`	フィールドターミネータ
`--ignore -i`	キーのには、りまれたをする
`--lines-terminated-by`	ターミネータをする
`--password -p`	パスワード
`--port -P`	
`--replace -r`	キーのはいエントリをきする
`--user -u`	ユーザー
`--where -w`	をする

`mysqlimport`は、インポートのテーブルをするために、をりいた、インポートされたファイルのをします。

Examples

な

タブでられたファイル`employee.txt`

1 `\t` Arthur Dent
2 `\t`マーヴィン
3 `\t` Zaphod Beeblebrox

```
$ mysql --user=user --password=password mycompany -e 'CREATE TABLE employee(id INT, name
VARCHAR(100), PRIMARY KEY (id))'

$ mysqlimport --user=user --password=password mycompany employee.txt
```

カスタムフィールドりをする

えられたテキストファイルemployee.txt

1 |アーサー·デント
2 |マーヴィン
3 | Zaphod Beeblebrox

```
$ mysqlimport --fields-terminated-by='|' mycompany employee.txt
```

カスタムのりをする

このは、ウィンドウのようなエンディングにです

```
$ mysqlimport --lines-terminated-by='\r\n' mycompany employee.txt
```

キーの

`Employee`テーブル

id	
3	Yooden Vranx

そして、 `employee.txt`ファイル

1 `\t` Arthur Dent
2 `\t`マーヴィン
3 `\t` Zaphod Beeblebrox

`--ignore`オプションはしたキーのエントリをします

```
$ mysqlimport --ignore mycompany employee.txt
```

id	
1	アーサー·デント
2	マーヴィン
3	Yooden Vranx

`--replace`オプションはいエントリをきする

```
$ mysqlimport --replace mycompany employee.txt
```

id	
1	アーサー・デント
2	マーヴィン
3	ザフォッドビーブロンズ

きインポート

```
$ mysqlimport --where="id>2" mycompany employee.txt
```

csvをインポートする

```
$ mysqlimport
    --fields-optionally-enclosed-by='"'
    --fields-terminated-by=,
    --lines-terminated-by="\r\n"
    mycompany employee.csv
```

オンラインでmysqlimportをむ https://riptutorial.com/ja/mysql/topic/5215/mysqlimport

15: MySQLクライアント

- mysql [オプション] [データベース]

パラメータ

パラメータ	
-D --database=name	データベースの
--delimiter=str	のりをします。デフォルトのものは ';'です。
-e --execute='command'	コマンド
-h --host=name	するホスト
-p --password=name	パスワード-pとパスワードのにスペースはありません
-p パスワードなし	パスワードがされます
-P --port=#	ポート
-s --silent	サイレントモードでは、がなくなります。セパレータとして\tをする
-ss	like -sとじですが、はします
-S --socket=path	ローカルインスタンスにするときにするソケットUnixまたはきパイプWindowsをする
--skip-column-names	をする
-u --user=name	ユーザー
-U --safe-updates --i-am-a-dummy	sql_safe_updates=ONをしてログインします。これにより、にキーをするDELETEとUPDATEのみがされます
-V --version	バージョンをしてする

Examples

ログイン

コマンドラインからMySQLにアクセスするには

```
mysql --user=username --password=pwd --host=hostname test_db
```

これはにすることができます

```
mysql -u username -p password -h hostname test_db
```

`password`をすると、MySQLはのとしてなパスワードをします。 `password`を`password` 、クライアントはあなたに「でない」というをします

```
mysql -u=username -p -h=hostname test_db
```

ローカルの`--socket` 、 - ソケットをしてソケットファイルをすことができます

```
mysql --user=username --password=pwd --host=localhost --socket=/path/to/mysqld.sock test_db
```

`socket`パラメーターをすると、クライアントはローカル·マシンのサーバーにしようとします。するにはサーバーがしているがあります。

コマンドをする

こののは、プロンプトをとせずに、またはスクリプトファイルにされたコマンドをするをしています。これは、シェルスクリプトがデータベースとするがあるににです。

からコマンドをする

```
$ mysql -uroot -proot test -e'select * from people'

+----+-------+--------+
| id | name  | gender |
+----+-------+--------+
|  1 | Kathy | f      |
|  2 | John  | m      |
+----+-------+--------+
```

をタブりのグリッドとしてフォーマットするには、-- `--silent`パラメータをします。

```
$ mysql -uroot -proot test -s -e'select * from people'

id      name    gender
1       Kathy   f
2       John    m
```

ヘッダーをするには

```
$ mysql -uroot -proot test -ss -e'select * from people'

1       Kathy   f
2       John    m
```

スクリプトファイルから

```
$ mysql -uroot -proot test < my_script.sql
```

```
$ mysql -uroot -proot test -e'source my_script.sql'
```

をファイルにきむ

```
$ mysql -uroot -proot test < my_script.sql > out.txt

$ mysql -uroot -proot test -s -e'select * from people' > out.txt
```

オンラインでMySQLクライアントをむ https://riptutorial.com/ja/mysql/topic/5619/mysqlクライアント

16: MySQLのパフォーマンスのヒント

Examples

ステートメントの

は、たちがクエリのをするのにつ、MySQLでされたクエリをいているのヒントです。

1. きなテーブルのどこでどこをするかによって、whereのカラムがインデックスかどうかをするがあります。 - から*をします。ここでuser_id> 2000. user_idがインデックスされている、クエリatlotのがスピードアップされます。は、およびキーのにもにです。

2. テーブルからのデータをするのではなく、よりさなセクションのコンテンツがなは、limitをしてみてください。むしろ、からEx - Select *をいてください。 lakhsからの20のだけがなは、limit Ex - Select *をLIMIT 20からしてください。

3. また、セットになをして、クエリをすることもできます。むしろ、からEx - Select *をいてください。テーブルにがたくさんあり、それらのうちのいくつかのデータしかたないは、データがなをするだけです。 - のID、をします。

4. whereでNULLをするためにしているは、Indexカラム。 SELECT * FROM tbl_name WHERE key_col IS NULLのようながある。 key_colがけされると、はよりにされます。

InnoDBテーブルのストレージレイアウトの

1. InnoDBでは、いPRIMARY KEYいをつの、またはいをするののいずれかをつと、くのディスクがになります。のキーは、じをすすべての2レコードにされます。キーがいは、AUTO_INCREMENTをキーとしてします。
2. CHARではなくVARCHARデータをして、をするか、NULLがいにします。 CHARNは、がくてもがNULLであっても、データをするためににNをします。テーブルがさくなると、バッファプールがくなり、ディスクI / Oがします。

   COMPACTデフォルトのInnoDBとutf8やsjisなどのセットをする、CHARNはサイズのをめますが、それでもNバイトです。

3. きな、またはするテキストまたはデータがまれているは、COMPRESSEDのをしてください。バッファー·プールにデータをちむため、またはフル·テーブル·スキャンをするためには、よりないディスクI / Oがです。なをすに、COMPRESSEDとCOMPACTのをしてできるをします。 ベンチマークは21よりもれていることはめったにありません。また、COMPRESSEDのbuffer_poolにはくのオーバーヘッドがあります。
4. データがしたサイズにするか、またはするテーブルがまたはメガバイトしたら、OPTIMIZE TABLEステートメントをしてテーブルをし、なスペースをしてください。されたは、フル·テーブル·スキャンをするためになディスクはなくなります。これは、ののやアプリケーシ

ョン･コードのチュ―ニングなどののがでないに、パフォ―マンスをさせるなです。 テ―ブルサイズにかかわらず、OPTIMIZE TABLEはめったにされません。これはコストがかかり、があるほどテ―ブルをすることはめったにないためです。 InnoDBは、B + Treesをたくさんのなスペ―スからしておくのがです。

OPTIMIZE TABLEは、のデ―タをコピ―し、をします。このは、のデ―タのパッキングがされ、およびディスクのがすることによるものです。は、テ―ブルのデ―タによってなります。のユ―ザ―にはきながあり、のユ―ザ―にはきながないことや、にテ―ブルをするまで、のとにがすることがあります。がであるや、リビルドのがバッファ―･プ―ルにまらない、このはくなるがあります。くのデ―タをテ―ブルにしたののは、しばしばのよりもはるかにです。

インデックスの

くの、インデックスはのをつインデックスよりもれたパフォ―マンスをします。なコンポジットインデックスをするには、このでをします。

- `= WHERE`ののの。 たとえば、 `WHERE a=12 AND b='xyz' ...` `INDEX(a,b,...) WHERE a=12 AND b='xyz' ...`
- `IN`S;オプティマイザはをびえることができます。
- 1つの "range"えば、 `x BETWEEN 3 AND 9` 、 `name LIKE 'J%'` ののをえてもしません。
- `GROUP BY`すべてのをに
- `ORDER BY`すべての。すべてが`ASC`か、またはすべてが`DESC`か、または8.0をしているにのみします。

と

- をしないでください。
- されないはスキップしてください。
- `WHERE`すべてのをしないは、 `GROUP BY`などにむはありません。
- `WHERE`して、 `ORDER BY`のみをけするとながあります。
- のを「す」ことはできません `DATE(x) = ...`インデックスでは`x`をできません。
- "のけえば、 `text_col(99)` は`text_col(99)`ないでしょう。つくことがあります。

とヒント

オンラインでMySQLのパフォ―マンスのヒントをむ

https://riptutorial.com/ja/mysql/topic/5752/mysqlのパフォ―マンスのヒント

17: MySQLのユニオン

- SELECT column_namesFROM table1 UNION SELECT column_namesFROMテーブル2;
- SELECT column_namesFROM table1 UNION ALL SELECT column_namesFROMテーブル2;
- SELECT column_namesFROM table1 WHERE col_name = "XYZ" UNION ALL SELECT column_namesFROM table2 WHERE col_name = "XYZ";

`UNION DISTINCT`じです`UNION`。 のため`UNION ALL`よりもいです。 いは、に`DISTINCT`または`ALL`ることです`DISTINCT`これによって、あなたはをすべきかをえています。

Examples

オペレータ

UNIONは、2つのSELECTステートメントのセット ののみ をするためにされます 。

クエリ「」テーブルと「」テーブルからなるのみをするには

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

```
Number of Records: 10

City
------
Aachen
Albuquerque
Anchorage
Annecy
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Bern
```

ユニオンALL

UNION ALLをして、「」テーブルと「」テーブルからすべてのもむをします。

クエリ

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
```

```
ORDER BY City;
```

```
Number of Records: 12

City
-------
Aachen
Albuquerque
Anchorage
Ann Arbor
Annecy
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Berlin
Bern
```

UNION ALLとWHERE

UNION ALLをして、ドイツのを「」テーブルと「」テーブルからすべてしますも。ここでは、`Country="Germany"` whereでします。

クエリ

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION ALL
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

```
Number of Records: 14
```

シティ	
アーヘン	ドイツ
ベルリン	ドイツ
ベルリン	ドイツ
ブランデンブルク	ドイツ
Cunewalde	ドイツ
クックスハーフェン	ドイツ
フランクフルト	ドイツ

フランクフルト	ドイツ
ケルン	ドイツ
ライプツィヒ	ドイツ
マンハイム	ドイツ
ミュンヘン	ドイツ
ミュンスター	ドイツ
シュトゥットガルト	ドイツ

オンラインでMySQLのユニオンをむ https://riptutorial.com/ja/mysql/topic/5376/mysqlのユニオン

18: MySQLのロックテーブル

- LOCK TABLESテーブル[READ |きます]; //テーブルをロックする
- UNLOCK TABLES; //テーブルをロックする

ロッキングはのをするためにされます。ロッキングはトランザクションをするにのみです。トランザクションは、まずデータベースからをみり、でそのをデータベースにきみます。の、、またはには、ロックはありません。

なロックには2あります

READ LOCK - ユーザーがテーブルからのみみりの。

WRITE LOCK - ユーザーがのみりときみのをっているとき。

ユーザーがに`WRITE LOCK`をしている、そのをのユーザーがみきすることはできません。ユーザーがテーブルに`READ LOCK`をすると、のユーザーも`READ LOCK`みったりしたりすることができますが、そのテーブルに`WRITE LOCK`をきみまたはすることはできません。

デフォルトのストレージエンジンがInnoDBの、MySQLはにレベルのロックをするため、のトランザクションがじテーブルをみきのためににすることができます。

InnoDBのすべてのストレージエンジンでは、MySQLはテーブルロックをします。

テーブルロックのはこちら

Examples

MySQLのロック

テーブルロックは*`ENGINE=MyISAM`*なツールですが、 *`ENGINE=InnoDB`*ほとんどにちません。 *InnoDB*でテーブルロックをするようにされているは、トランザクションのをするがあります。

MySQLは、のセッションとしてテーブルにアクセスするで、またはセッションがアクセスをとするにのセッションがテーブルをすることをぐで、クライアントセッションでテーブルロックをにできます。セッションは、のロックのみをまたはできます。 1つのセッションはのセッションのロックをしたり、のセッションがするロックをすることはできません。

ロックをすると、トランザクションをエミュレートしたり、テーブルをするときにをさせることができます。については、このセクションのでします。

コマンド `LOCK TABLES table_name READ|WRITE;`

1つのにロック·タイプのみをりてることができます。

READ LOCK

```
LOCK TABLES table_name READ;
```

WRITE LOCK

```
LOCK TABLES table_name WRITE;
```

ロックがされているかどうかをするには、のコマンドをします。

```
SHOW OPEN TABLES;
```

すべてのロックをフラッシュ/するには、のコマンドをします。

```
UNLOCK TABLES;
```

```
LOCK TABLES products WRITE:
INSERT INTO products(id,product_name) SELECT id,old_product_name FROM old_products;
UNLOCK TABLES;
```

のでは、はテ―ブルのロックをするまでテ―ブルにデ―タをきむことができません

```
LOCK TABLES products READ:
INSERT INTO products(id,product_name) SELECT id,old_product_name FROM old_products;
UNLOCK TABLES;
```

のでは、はテ―ブルのロックをするまでテ―ブルからデ―タをみることができません

レベルのロック

テ―ブルがInnoDBをしている、MySQLはにレベルのロックをし、のトランザクションがじテ―ブルをみきのためににできるようにします。

2つのトランザクションがじをしようとしていて、ともレベルのロックをしている、トランザクションの1つはもうのトランザクションがするまでします。

レベルのロックは、がなごとに`SELECT ... FOR UPDATE`をしてもできます。

レベルのロックをにする2つのをしてください

1

```
START TRANSACTION;
SELECT ledgerAmount FROM accDetails WHERE id = 1 FOR UPDATE;
```

1では、 `SELECT ... FOR UPDATE`によってされたレベルのロックです。

2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1;
```

2でじをしようとすると、1でトランザクションがするのをつか、 `innodb_lock_wait_timeout`にってエラーメッセージがされます。デフォルトは50です。

```
Error Code: 1205. Lock wait timeout exceeded; try restarting transaction
```

このロックのをするには、 `SHOW ENGINE INNODB STATUS`します

```
---TRANSACTION 1973004, ACTIVE 7 sec updating
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 360, 1 row lock(s)
MySQL thread id 4, OS thread handle 0x7f996beac700, query id 30 localhost root update
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1
------- TRX HAS BEEN WAITING 7 SEC FOR THIS LOCK TO BE GRANTED:
```

2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 250 WHERE id=2;
1 row(s) affected
```

しかし、2ののをするにエラーがすることはありません。

1

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 750 WHERE id=1;
COMMIT;
1 row(s) affected
```

トランザクションは1でコミットされるため、ロックはされます。

2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1;
1 row(s) affected
```

Connection 1がトランザクションをしてロックをした、Connection 2でエラーなしでがされます。

オンラインでMySQLのロックテーブルをむ https://riptutorial.com/ja/mysql/topic/5233/mysqlのロックテーブル

19: MySQL

Examples

ルートパスワードをする

```
mysqladmin -u root -p'old-password' password 'new-password'
```

データベースの

スクリプトをってすべてのテーブルをし、データベースをするのにです

```
mysqladmin -u[username] -p[password] drop [database]
```

なをってしてください。

データベースをSQLスクリプトとしてDROPするにはデータベースにDROPがです

```
DROP DATABASE database_name
```

または

```
DROP SCHEMA database_name
```

アトミックRENAMEテーブルリロード

```
RENAME TABLE t TO t_old, t_copy TO t;
```

RENAME TABLEがされているは、するテーブルにのセッションでアクセスすることはできません。したがって、のははののをけません。

Atomic Renameは、にDELETEとロードがするのをつことなくテーブルをにロードするためのものです。

```
CREATE TABLE new LIKE real;
load `new` by whatever means - LOAD DATA, INSERT, whatever
RENAME TABLE real TO old, new TO real;
DROP TABLE old;
```

オンラインでMySQLをむ https://riptutorial.com/ja/mysql/topic/2991/mysql

20: ORDER BY

Examples

コンテキスト

SELECTのにはのがあります。

```
SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ...
    ORDER BY ...  -- goes here
    LIMIT ... OFFSET ...;

( SELECT ... ) UNION ( SELECT ... ) ORDER BY ...  -- for ordering the result of the UNION.

SELECT ... GROUP_CONCAT(DISTINCT x ORDER BY ... SEPARATOR ...) ...

ALTER TABLE ... ORDER BY ... -- probably useful only for MyISAM; not for InnoDB
```

ベーシック

ORDER BY x

xはのデータです。

- NULLsはNULLsのものにします。
- デフォルトはASC もいものからもいもの
- COLLATION VARCHARなどは、のCOLLATIONにってCOLLATIONられます
- ENUMsは、のにべられます。

ASCending / DESCending

```
ORDER BY x ASC  -- same as default
ORDER BY x DESC  -- highest to lowest
ORDER BY lastname, firstname  -- typical name sorting; using two columns
ORDER BY submit_date DESC  -- latest first
ORDER BY submit_date DESC, id ASC  -- latest first, but fully specifying order.
```

- ASC = ASCENDING 、 DESC = DESCENDING
- DESCでもNULLsがにます。
- のでは、 INDEX(x) 、 INDEX(lastname, firstname) 、 INDEX(submit_date)はパフォーマンスをにさせるがあります。

しかし、...ののようにASCとDESCさせることは、インデックスをしてをることはできません。
INDEX(submit_date DESC, id ASC)けもありません - " DESC "はINDEXでにされますがされます。

いくつかのトリック

```
ORDER BY FIND_IN_SET(card_type, "MASTER-CARD,VISA,DISCOVER") -- sort 'MASTER-CARD' first.
ORDER BY x IS NULL, x  -- order by `x`, but put `NULLs` last.
```

カスタムオーダー

```
SELECT * FROM some_table WHERE id IN (118, 17, 113, 23, 72)
ORDER BY FIELD(id, 118, 17, 113, 23, 72);
```

されたIDのでをします。

id	...
118	...
17	...
113	...
23	...
72	...

IDがすでにソートされていて、をするがあるにです。

オンラインでORDER BYをむ https://riptutorial.com/ja/mysql/topic/5469/order-by

21: Prepared StatementをしたUn-Pivotテーブル

Examples

についてのセットのピボットをする

のは、BI /レポートので、トランザクションデータをピボットされていないデータにしようとしているときににです。ピボットされないディメンションにはなセットがあります。

このでは、データテーブルにマークされたのデータがまれているとします。

データテーブルはのとおりです。

```
create table rawdata

(
 PersonId VARCHAR(255)
,Question1Id INT(11)
,Question2Id INT(11)
,Question3Id INT(11)
)
```

rawdataテーブルは、ETLプロシージャのとしてのテーブルであり、さまざまなのをつことができます。は、のの、つまり、になるにしてじをすることです。

は、rawdataテーブルのおもちゃのです

PersonId	Question1Id	Question2Id	Question3Id
Giannaros	1	3	1
Patra	2	4	3

データをunpivotするためのよくられているなは、UNION ALLをすることです。

```
create table unpivoteddata

(

 PersonId VARCHAR(255)
,QuestionId VARCHAR(255)
,QuestionValue INT(11)

);

INSERT INTO unpivoteddata SELECT PersonId, 'Question1Id' col, Question1Id
FROM rawdata
UNION ALL
SELECT PersonId, 'Question2Id' col, Question2Id
FROM rawdata
```

```
UNION ALL
SELECT PersonId, 'Question3Id' col, Question3Id
FROM rawdata;
```

ここでは、ののQuestionIdをピボットするをしたいとえています。そのためには、のをにするをするがあります。どのをピボットするがあるかをできるようにするために、GROUP_CONCATをし、デ―タが 'int'にされているをします。 GROUP_CONCATには、されるSELECTのすべてのもまれています。

```
set @temp2 = null;

SELECT GROUP_CONCAT(' SELECT ', 'PersonId',',','''',COLUMN_NAME,'''', ' col
',',',COLUMN_NAME,' FROM rawdata' separator ' UNION ALL' ) FROM     INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'rawdata' AND DATA_TYPE = 'Int' INTO     @temp2;

select @temp2;
```

のでは、がパタ―ンにするをすることができました。たとえば、

```
DATA_TYPE = 'Int'
```

つかいます

```
COLUMN_NAME LIKE 'Question%'
```

またはETLフェ―ズでできるなものでなければなりません。

されたステ―トメントはのようにされます。

```
set @temp3 = null;

select concat('INSERT INTO unpivoteddata',@temp2) INTO @temp3;

select @temp3;

prepare stmt FROM @temp3;
execute stmt;
deallocate prepare stmt;
```

unpivoteddataテ―ブルはのとおりです。

```
SELECT * FROM unpivoteddata
```

PersonId	QuestionId	QuestionValue
Giannaros	Question1Id	1
Patra	Question1Id	2
Giannaros	Question2Id	3
Patra	Question2Id	4
Giannaros	Question3Id	1
Patra	Question3Id	3

にってをし、されたステートメントをすることは、データをにピボットするなです。

オンラインでPrepared StatementをしたUn-Pivotテーブルをむ
https://riptutorial.com/ja/mysql/topic/6491/prepared-statementをしたun-pivotテーブル

22: PREPAREステートメント

- PREPARE stmt_name FROM preparable_stmt
- EXECUTE stmt_name [USING @var_name [、@var_name] ...]
- {DEALLOCATE | DROP} PREPARE stmt_name

Examples

PREPARE、EXECUTEおよびDEALLOCATE PREPAREステートメント

PREPAREはのためのをする

EXECUTEはプリペアドステートメントをします。

DEALLOCATE PREPAREはされたをする

```
SET @s = 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
PREPARE stmt2 FROM @s;
SET @a = 6;
SET @b = 8;
EXECUTE stmt2 USING @a, @b;
```

```
+------------+
| hypotenuse |
+------------+
|         10 |
+------------+
```

に、

```
DEALLOCATE PREPARE stmt2;
```

ノート

- @variablesをするがあります。FROM @sのDECLAREはしないで`FROM @s`
- Prepareなどのなは、テーブルのなど、バインディングがしないのクエリを「する」ことです。

と

これは、するをしていのである`SELECT`して`CONCAT`、それを+した@variablesのをしてください-。それはきないになり、そしてそれはかでありますをむがする。

をしてテーブルをする

```
SET v_column_definition := CONCAT(
  v_column_name
  ,' ',v_column_type
  ,' ',v_column_options
);

SET @stmt := CONCAT('ALTER TABLE ADD COLUMN ', v_column_definition);

PREPARE stmt FROM @stmt;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

オンラインでPREPAREステートメントをむ https://riptutorial.com/ja/mysql/topic/2603/prepareステートメント

23: PS1をカスタマイズする

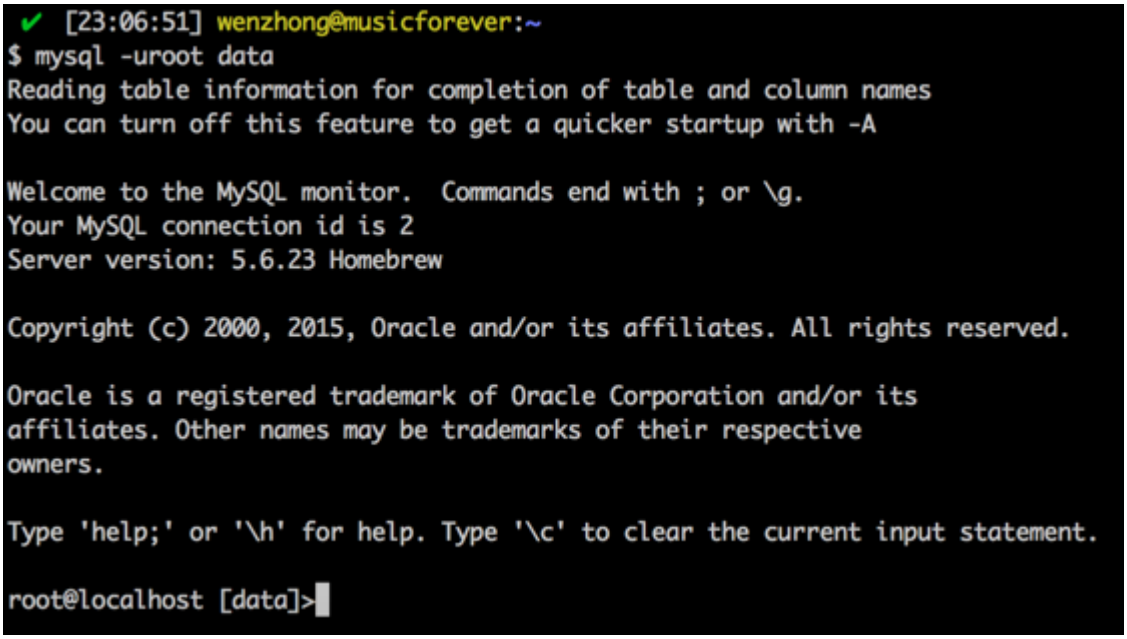
Examples

のデータベースでMySQL PS1をカスタマイズする

.bashrcまたは.bash_profileに、をします。

```
export MYSQL_PS1="\u@\h [\d]>"
```

MySQLクライアントPROMPTにのuser @ host [database]をさせます。



MySQLコンフィギュレーションファイルによるカスタムPS1

`mysqld.cnf`またはのもの

```
[mysql]
prompt = '\u@\h [\d]> '
```

これにより、 `.bashrc`をすることなく、のがられ`.bashrc`。

オンラインでPS1をカスタマイズするをむ https://riptutorial.com/ja/mysql/topic/5795/ps1をカスタマイズする

24: SSLのセットアップ

Examples

Debianベースのシステムのセットアップ

これは、MySQLがインストールされており、 sudoがされていることをとしています。

CAおよびSSLキーの

OpenSSLとライブラリがインストールされていることをしてください

```
apt-get -y install openssl
apt-get -y install libssl-dev
```

に、SSLファイルのディレクトリをしてします。

```
mkdir /home/ubuntu/mysqlcerts
cd /home/ubuntu/mysqlcerts
```

をするには、にするCAをします。

```
openssl genrsa 2048 > ca-key.pem
openssl req -new -x509 -nodes -days 3600 -key ca-key.pem -out ca.pem
```

プロンプトでしたは、にしません。に、サーバーのをし、のCAをしてします。

```
openssl req -newkey rsa:2048 -days 3600 -nodes -keyout server-key.pem -out server-req.pem
openssl rsa -in server-key.pem -out server-key.pem

openssl x509 -req -in server-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -
out server-cert.pem
```

に、クライアントのキーをします。

```
openssl req -newkey rsa:2048 -days 3600 -nodes -keyout client-key.pem -out client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -req -in client-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -
out client-cert.pem
```

すべてがしくされていることをするには、キーをします。

```
openssl verify -CAfile ca.pem server-cert.pem client-cert.pem
```

MySQLにをする

MySQLファイルをきます 。えば

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

`[mysqld]`セクションのに、のオプションをします。

```
ssl-ca = /home/ubuntu/mysqlcerts/ca.pem
ssl-cert = /home/ubuntu/mysqlcerts/server-cert.pem
ssl-key = /home/ubuntu/mysqlcerts/server-key.pem
```

MySQLをします。えば

```
service mysql restart
```

SSLをテストする

じでし、されたクライアントをして、オプションの`ssl-ca` 、 `ssl-cert` 、および`ssl-key`をします。たとえば、 `cd /home/ubuntu/mysqlcerts`ます。

```
mysql --ssl-ca=ca.pem --ssl-cert=client-cert.pem --ssl-key=client-key.pem -h 127.0.0.1 -u
superman -p
```

ログイン、がにであることをします。

```
superman@127.0.0.1 [None]> SHOW VARIABLES LIKE '%ssl%';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| have_openssl  | YES                                     |
| have_ssl      | YES                                     |
| ssl_ca        | /home/ubuntu/mysqlcerts/ca.pem          |
| ssl_capath    |                                         |
| ssl_cert      | /home/ubuntu/mysqlcerts/server-cert.pem |
| ssl_cipher    |                                         |
| ssl_crl       |                                         |
| ssl_crlpath   |                                         |
| ssl_key       | /home/ubuntu/mysqlcerts/server-key.pem  |
+---------------+-----------------------------------------+
```

のもチェックできます

```
superman@127.0.0.1 [None]> STATUS;
...
SSL:                    Cipher in use is DHE-RSA-AES256-SHA
...
```

SSLの

これは、 `REQUIRE SSL`をして`GRANT`をしてわれます。

```
GRANT ALL PRIVILEGES ON *.* TO 'superman'@'127.0.0.1' IDENTIFIED BY 'pass' REQUIRE SSL;
FLUSH PRIVILEGES;
```

、 `superman` は SSLでするがあります。

クライアントをしないは、のクライアントをして、それをすべてのクライアントににします。
MySQLのファイルをきます。

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

`[client]`セクションで、のオプションをします。

```
ssl-ca = /home/ubuntu/mysqlcerts/ca.pem
ssl-cert = /home/ubuntu/mysqlcerts/client-cert.pem
ssl-key = /home/ubuntu/mysqlcerts/client-key.pem
```

`superman`は、SSLをしてログインするためにのようにするだけです

```
mysql -h 127.0.0.1 -u superman -p
```

Pythonなどののプログラムからするは、、connectのパラメータがです。 Pythonの

```
import MySQLdb
ssl = {'cert': '/home/ubuntu/mysqlcerts/client-cert.pem', 'key':
'/home/ubuntu/mysqlcerts/client-key.pem'}
conn = MySQLdb.connect(host='127.0.0.1', user='superman', passwd='imsoawesome', ssl=ssl)
```

およびさらなる

- https://www.percona.com/blog/2013/06/22/setting-up-mysql-ssl-and-secure-connections/
- https://lowendbox.com/blog/getting-started-with-mysql-over-ssl/
- http://xmodulo.com/enable-ssl-mysql-server-client.html
- https://ubuntuforums.org/showthread.php?t=1121458

CentOS7 / RHEL7のセットアップ

このでは、2つのサーバーをしています。

1. dbserverデータベースがする
2. appclientアプリケーションがどこにあるか

*FWIW*では、のサーバが*SELinux*をしています。

まず、dbserverにログオンします。

をするためのディレクトリをします。

```
mkdir /root/certs/mysql/ && cd /root/certs/mysql/
```

サーバーをする

```
openssl genrsa 2048 > ca-key.pem
openssl req -sha1 -new -x509 -nodes -days 3650 -key ca-key.pem > ca-cert.pem
openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout server-key.pem > server-req.pem
openssl rsa -in server-key.pem -out server-key.pem
openssl x509 -sha1 -req -in server-req.pem -days 730  -CA ca-cert.pem -CAkey ca-key.pem -
set_serial 01 > server-cert.pem
```

サーバーを/ etc / pki / tls / certs / mysql /

ディレクトリパスはCentOSまたはRHELとみなされますのディストリビューションではにじてします。

```
mkdir /etc/pki/tls/certs/mysql/
```

フォルダとファイルにするアクセスをしてください。 mysqlにはなとアクセスがです。

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

MySQL / MariaDBをする

```
# vi /etc/my.cnf
# i
[mysqld]
bind-address=*
ssl-ca=/etc/pki/tls/certs/ca-cert.pem
ssl-cert=/etc/pki/tls/certs/server-cert.pem
ssl-key=/etc/pki/tls/certs/server-key.pem
# :wq
```

その、

```
systemctl restart mariadb
```

ファイアウォールをいてappclientからのをすることをれないでくださいIP 1.2.3.4を

```
firewall-cmd --zone=drop --permanent --add-rich-rule 'rule family="ipv4" source
address="1.2.3.4" service name="mysql" accept'
# I force everything to the drop zone.  Season the above command to taste.
```

すぐfirewalldをする

```
service firewalld restart
```

に、dbserverのmysqlサーバにログインします。

```
mysql -uroot -p
```

をして、クライアントのユーザーをします。 GRANTでSSLをしてください。

```
GRANT ALL PRIVILEGES ON *.* TO 'iamsecure'@'appclient' IDENTIFIED BY 'dingdingding' REQUIRE
SSL;
FLUSH PRIVILEGES;
# quit mysql
```

のステップから/ root / certs / mysqlにいなければなりません。そうでないは、のいずれかのコマンドをしてcdにってください。

クライアントをする

```
openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout client-key.pem > client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -sha1 -req -in client-req.pem -days 730 -CA ca-cert.pem -CAkey ca-key.pem -
set_serial 01 > client-cert.pem
```

は、サーバーとクライアントのにじをしました。 YMMV。

このコマンドではまだ/ *root / certs / mysql /*であることをしてください

サーバーとクライアントのCAを1つのファイルにする

```
cat server-cert.pem client-cert.pem > ca.pem
```

2つのがされることをしてください。

```
cat ca.pem
```

すぐサーバーののわり。

のをき、

```
ssh appclient
```

とじように、クライアントのなホームをする

```
mkdir /etc/pki/tls/certs/mysql/
```

に、クライアントdbserverでをappclientにします。あなたはそれらをscpしたり、ファイルをつずつコピーしてりけたりすることができます。

```
scp dbserver
# copy files from dbserver to appclient
# exit scp
```

、フォルダとファイルにするアクセスをしてください。 mysqlにはなとアクセスがです。

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

3つのファイルがです。ファイルは、ユーザmysqlがしています。

```
/etc/pki/tls/certs/mysql/ca.pem
/etc/pki/tls/certs/mysql/client-cert.pem
/etc/pki/tls/certs/mysql/client-key.pem
```

`[client]`セクションでappclientのMariaDB / MySQLをし`[client]`。

```
vi /etc/my.cnf
# i
[client]
ssl-ca=/etc/pki/tls/certs/mysql/ca.pem
ssl-cert=/etc/pki/tls/certs/mysql/client-cert.pem
ssl-key=/etc/pki/tls/certs/mysql/client-key.pem
# :wq
```

appclientのmariadbサービスをします。

```
systemctl restart mariadb
```

まだここのクライアントに

これがされるはずですssl TRUE

```
mysql --ssl --help
```

さて、appclientのmysqlインスタンスにログインしてください

```
mysql -uroot -p
```

ののにYESとされるはずです

```
show variables LIKE '%ssl';
    have_openssl    YES
    have_ssl            YES
```

にはた

```
 have_openssl NO
```

mariadb.logのをらかにした

    SSLエラー '/etc/pki/tls/certs/mysql/client-cert.pem'からをできません

は、rootがしていたclient-cert.pemとそのフォルダをむことでした。は、/ etc / pki / tls / certs / mysql /のをmysqlにすることでした。

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

にじて、すぐのステップからmariadbをします。

すぐなをテストするができています

たちはまだここにappclientにいる

でしたアカウントをしてdbserverのmysqlインスタンスにしようとします。

```
mysql -h dbserver -u iamsecure -p
# enter password dingdingding (hopefully you changed that to something else)
```

しがあれば、いなくログインするがあります。

SSLをにしてしていることをするには、MariaDB / MySQLプロンプトからのコマンドをします。

```
\s
```

それはバックスラッシュs、ステータスです

のステータスがされます。はのようになります。

```
Connection id:          4
Current database:
Current user:           iamsecure@appclient
SSL:            Cipher in use is DHE-RSA-AES256-GCM-SHA384
Current pager:          stdout
Using outfile:          ''
Using delimiter:    ;
Server:             MariaDB
Server version:         5.X.X-MariaDB MariaDB Server
Protocol version:   10
Connection:         dbserver via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
```

```
Client characterset:    utf8
Conn.  characterset:    utf8
TCP port:        3306
Uptime:            42 min 13 sec
```

のでエラーがされなくなったは、のGRANTステートメントをべて、いのあるやがないことをしてください。

SSLエラーがしたは、このガイドにってがであることをしてください。

これはRHEL7でし、おそらくCentOS7でもします。これらのながのでするかどうかはできません。

これはかがしとをすることをっています。

オンラインでSSLのセットアップをむ https://riptutorial.com/ja/mysql/topic/7563/sslのセットアップ

25: VIEW

- CREATE VIEW view_name AS SELECT column_namesFROM table_name WHEREです。///シンプルなビュ—の
- CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED |マージ| TEMPTABLE}] [DEFINER = {user | CURRENT_USER}] [SQLセキュリティ| DEFINER | INVOKER}] VIEWビュ—[リスト] AS select_statement [WITH [CASCADED |ローカル] CHECK OPTION]; ///なビュ—の
- DROP VIEW [IF EXISTS] [db_name。]ビュ—; ///ドロップビュ—の

パラ メ—タ—

パラ メ—タ—	
ビュ—	ビュ—の
SELECTステ—トメント	ビュ—にパックされるSQLステ—トメント1つのテ—ブルからデ—タをフェッチするSELECTにすることができます。

ビュ—はテ—ブルであり、されるデ—タはまれません。なせをももくことをけることができます。

- ビュ—がされるに、そのはに`SELECT`ステ—トメントからされます。 `SELECT`にFROMにサブクエリをめることはできません。
- ビュ—がされると、テ—ブルのようにがされ、テ—ブルのように`SELECT`できます。

テ—ブルのいくつかのをのユ—ザ—からしたいは、ビュ—をするがあります。

- たとえば、では、マネ—ジャが「Sales」というのテ—ブルからをほとんどしないようにしたいが、ソフトウェアエンジニアがテ—ブル「Sales」のすべてのフィ—ルドをするはない。ここでは、マネ—ジャとソフトウェアエンジニアのために2つのなるビュ—をできます。

パフォ—マンス 。 `VIEWs`はなです。ただし、ビュ—のがりまれたのクエリよりパフォ—マンスがいもあれば、いもあります。オプティマイザはこれを「りみ」しようとしますが、ずしもするとはりません。 MySQL 5.7.6ではオプティマイザのがさらにされました。しかし、 `VIEW`をしても、せをすることはできません。

Examples

ビュ—をする

CREATE VIEWには、ビュ—にするCREATE VIEWと、SELECTでされたにするがです。

SELECTステートメントののでされるのは、SELECTがです。 OR REPLACEがするは、ビュー にするDROPもです。このセクションのでするように、CREATE VIEWにはDEFINERにじて SUPERがながあります。

ビューがされると、チェックがわれます。

ビューはデータベースにします。デフォルトでは、デフォルトのデータベースにしいビューがされます。のデータベースににビューをするには、

えば

db_name.view_name

```
mysql> CREATE VIEW test.v AS SELECT * FROM t;
```

- データベースでは、ベーステーブルとビューはじをするので、ベーステーブルとビューはじをつことはできません。

VIEWはのことができます

- くののSELECTからする
- ベーステーブルまたはのビューを
- 、UNION、およびサブクエリをする
- SELECTはテーブルをするはありません

もうつの

のでは、のテーブルから2つのをするビューと、それらのからされたをしています。

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
+------+-------+-------+
```

- MySQL 5.7.7よりでは、SELECTにFROMにサブクエリをめることはできません。
- SELECTステートメントは、システムまたはユーザーをすることはできません。
- ストアド·プログラムで、SELECTステートメントはプログラム·パラメータまたはローカルをすることはできません。
- SELECTステートメントは、プリペアドステートメントパラメータをすることはできません。
- にされているテーブルまたはビューがするがあります。ビューがされた、テーブルまたはビューをドロップすることができます。
  はをします。この、ビューをするとエラーがします。こののビューをチェックするには、

CHECK TABLEステートメントをします。
- はTEMPORARYテーブルをすることはできません。
  TEMPORARYビューをします。
- トリガーをビューにけることはできません。
- SELECTののエイリアスは、エイリアスではなく、64のとされます
  256のさ。
- `VIEW`は、の`SELECT`とにすることもしないこともできます。それにすることはまずありません
  。

2つのテーブルからのビュー

ビューは、のからデータをりむためにできるにもです。

```
CREATE VIEW myview AS
SELECT a.*, b.extra_data FROM main_table a
LEFT OUTER JOIN other_table b
ON a.id = b.id
```

MySQLビューでは、ビューはマテリアライズされません。 `SELECT * FROM myview`というなクエリをすると、mysqlはにシーンのろにLEFT JOINをします。

されたビューは、のビューやテーブルにすることができます

VIEWによるの

`VIEW`はテーブルのようににします。あなたはできますが`UPDATE`を、あなたは、またはそのテーブルにビューをすることができないがあります。に、ビューの`SELECT`がをとするほどな、 `UPDATE`はされません。

`GROUP BY`、 `UNION`、 `HAVING`、 `DISTINCT`、およびいくつかのサブクエリのようなものは、ビューをできないようにします。 リファレンスマニュアルの。

ビューをする

- のデータベースにビューをしてします。

```
CREATE VIEW few_rows_from_t1 AS SELECT * FROM t1 LIMIT 10;
DROP VIEW few_rows_from_t1;
```

- なるデータベースのテーブルをするビューをしてする。

```
CREATE VIEW table_from_other_db AS SELECT x FROM db1.foo WHERE x IS NOT NULL;
DROP VIEW table_from_other_db;
```

オンラインでVIEWをむ https://riptutorial.com/ja/mysql/topic/1489/view

26: イベント

Examples

イベントをする

Mysqlには、されているもののくがSQLにし、ファイルにすることがないときに、なcronのやりとりをけるためのEVENTがあります。マニュアルペ―ジをしてくださいここに 。イベントをなでするようにスケジュ―ルされたストアドプロシ―ジャとえてください。

イベントののデバッグにをするには、イベントをするためにグロ―バル·イベント·ハンドラをオンにするがあることにしてください。

```
SHOW VARIABLES WHERE variable_name='event_scheduler';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | OFF   |
+-----------------+-------+
```

オフにすると、もトリガ―されません。だからそれをオンにする

```
SET GLOBAL event_scheduler = ON;
```

テストのスキ―マ

```
create table theMessages
(   id INT AUTO_INCREMENT PRIMARY KEY,
    userId INT NOT NULL,
    message VARCHAR(255) NOT NULL,
    updateDt DATETIME NOT NULL,
    KEY(updateDt)
);

INSERT theMessages(userId,message,updateDt) VALUES (1,'message 123','2015-08-24 11:10:09');
INSERT theMessages(userId,message,updateDt) VALUES (7,'message 124','2015-08-29');
INSERT theMessages(userId,message,updateDt) VALUES (1,'message 125','2015-09-03 12:00:00');
INSERT theMessages(userId,message,updateDt) VALUES (1,'message 126','2015-09-03 14:00:00');
```

のインサ―トは、をすためにされています。でされる2つのイベントはをすることにしてください。

2イベントを、1、10ごとに2

らがにやっていることをするいにしてふ。ポイントはとスケジュ―ルです。

```
DROP EVENT IF EXISTS `delete7DayOldMessages`;
DELIMITER $$
CREATE EVENT `delete7DayOldMessages`
  ON SCHEDULE EVERY 1 DAY STARTS '2015-09-01 00:00:00'
  ON COMPLETION PRESERVE
DO BEGIN
   DELETE FROM theMessages
   WHERE datediff(now(),updateDt)>6; -- not terribly exact, yesterday but <24hrs is still 1
day

  -- Other code here

END$$
DELIMITER ;
```

...

```
DROP EVENT IF EXISTS `Every_10_Minutes_Cleanup`;
DELIMITER $$
CREATE EVENT `Every_10_Minutes_Cleanup`
  ON SCHEDULE EVERY 10 MINUTE STARTS '2015-09-01 00:00:00'
  ON COMPLETION PRESERVE
DO BEGIN
   DELETE FROM theMessages
   WHERE TIMESTAMPDIFF(HOUR, updateDt, now())>168; -- messages over 1 week old (168 hours)

   -- Other code here
END$$
DELIMITER ;
```

イベントのをするなる

```
SHOW EVENTS FROM my_db_name; -- List all events by schema name (db name)
SHOW EVENTS;
SHOW EVENTS\G; -- <--------- I like this one from mysql> prompt

*************************** 1. row ***************************
                  Db: my_db_name
                Name: delete7DayOldMessages
             Definer: root@localhost
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: 1
      Interval field: DAY
              Starts: 2015-09-01 00:00:00
                Ends: NULL
              Status: ENABLED
          Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: utf8_general_ci
*************************** 2. row ***************************
                  Db: my_db_name
                Name: Every_10_Minutes_Cleanup
             Definer: root@localhost
           Time zone: SYSTEM
```

```
               Type: RECURRING
         Execute at: NULL
      Interval value: 10
      Interval field: MINUTE
             Starts: 2015-09-01 00:00:00
               Ends: NULL
             Status: ENABLED
         Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: utf8_general_ci
2 rows in set (0.06 sec)
```

するランダムなもの

`DROP EVENT someEventName;` - イベントとそのコーをする

`ON COMPLETION PRESERVE` - イベントがされたら、します。それのはされます。

ィベントはトリガーのようなものです。ユーザーのプログラムによってびされることはありません。むしろ、らはされている。そのように、らはかにするかする。

ページへのリンクは、にすように、のがかなりになっています。

```
quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
          WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
          DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

イベントは、システムのなタスクやスケジュールされたタスクをするなメカニズムです。それらには、くの、DDLおよびDMLルーチン、およびにむかもしれないなをめることができます。 ストアドプログラムにするというMySQLのマニュアルページをしてください。

オンラインでイベントをむ https://riptutorial.com/ja/mysql/topic/4319/イベント

27: インサート

1. INSERT [LOW_PRIORITY |れて| [パーティション、...] [col_name、...] {VALUES | VALUE} {expr | DEFAULT}、...、.....、... [キーではcol_name = expr [、col_name = expr] ...]
2. INSERT [LOW_PRIORITY |れて| HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITIONパーティション、...] SET col_name = {expr | DEFAULT}、... [キーでは col_name = expr [、col_name = expr] ...]
3. INSERT [LOW_PRIORITY | col_name、...] SELECT ... [キーにcol_name = expr [、col_name = expr] ...]
4. exprは、リストでにされたをできます。たとえば、col2のがにりてられていたcol1をするため、これをうことができます。
   INSERT INTO tbl_namecol1、col2VALUES15、col1 * 2;
5. VALUESをするINSERTステートメントは、のをできます。これをうには、でまれカンマでられたののリストをめます。
   INSERT INTO tbl_namea、b、cVALUES1,2,3、4,5,6、7,8,9;
6. のリストは、かっこでむがあります。リストののがのとしないため、のはです。
   INSERT INTO tbl_namea、b、cVALUES1,2,3,4,5,6,7,8,9;
7. INSERT ... SELECT
   INSERT [LOW_PRIORITY | [col_name、...] SELECT ... [キーcol_name = expr、...]
8. INSERT ... SELECTをすると、1つまたはのからくのをにすばやくできます。えば
   INSERT INTO tbl_temp2fld_idSELECT tbl_temp1.fld_order_id FROM tbl_temp1 WHERE tbl_temp1.fld_order_id> 100;

のINSERT

Examples

```
INSERT INTO `table_name` (`field_one`, `field_two`) VALUES ('value_one', 'value_two');
```

このなでは、 `table_name`はデータをする、 `field_one`と`field_two`はデータをするフィールド、`value_one`と`value_two`はそれぞれ`field_one`と`field_two`にしてうデータです。

コードにデータをするフィールドをリストすることは、テーブルがされ、しいがされた、がそこにしないにするようにすることをおめします

デュプリケートキーでINSERT

```
INSERT INTO `table_name`
    (`index_field`, `other_field_1`, `other_field_2`)
    VALUES
    ('index_value', 'insert_value', 'other_value')
    ON DUPLICATE KEY UPDATE
        `other_field_1` = 'update_value',
        `other_field_2` = VALUES(`other_field_2`);
```

これにより、されたが`table_name` `INSERT`されますが、ユニークキーがすでにするは、`other_field_1`がしいにされます。
ときどきキーをすると、をするわりに`INSERT`にされたのにアクセスするために`VALUES()`をするとです。これにより、 `INSERT`と`UPDATE`をしてなるをできます。のをしてください`other_field_1`するようにされている`insert_value`に`INSERT`またはする`update_value`に`UPDATE`ながら`other_field_2`ににされている`other_value`。

キーIODKUがするためになのは、したをするユニークなキーをむスキーマです。このユニークキーは、プライマリキーであってもなくてもよい。これは、のまたはキーののキーにすることができます。

のをする

```
INSERT INTO `my_table` (`field_1`, `field_2`) VALUES
    ('data_1', 'data_2'),
    ('data_1', 'data_3'),
    ('data_4', 'data_5');
```

これは、1つの`INSERT`ステートメントでのをにするなです。

こののバッチインサートは、を1つずつするよりもはるかにです。に、このデで1のに100をするのは、それらをすべてにするの10です。

のをする

なデータセットをインポートする、のでは、のはキーのなどによってクエリがするをスキップすることがましいがあります。これは、 `INSERT IGNORE`をしてうことができ`INSERT IGNORE`。

ののデータベースをしてください。

```
SELECT * FROM `people`;
--- Produces:
+----+------+
| id | name |
+----+------+
|  1 | john |
|  2 | anna |
+----+------+


INSERT IGNORE INTO `people` (`id`, `name`) VALUES
    ('2', 'anna'), --- Without the IGNORE keyword, this record would produce an error
```

```
    ('3', 'mike');

SELECT * FROM `people`;
--- Produces:
+----+--------+
| id |  name  |
+----+--------+
|  1 |  john  |
|  2 |  anna  |
|  3 |  mike  |
+----+--------+
```

なことは、 *INSERT IGNORE*もかにのエラーをスキップすることです.Mysqlのドキュメントにはのようにかれています。

> IGNOREがされていない、エラーをきこすデータによってがされます。 IGNOREでは、ながもいにされ、>されます。はされますが、はされません。

- のセクションはのためにされていますが、ベストプラクティスとはみなされませんたとえば、のがテーブルにされたなど。

のすべてのにするのをすると、のようにINSERTのリストをできます。

```
INSERT INTO `my_table` VALUES
        ('data_1', 'data_2'),
        ('data_1', 'data_3'),
        ('data_4', 'data_5');
```

INSERT SELECTのテーブルからのデータの

これは、SELECTステートメントでのテーブルのデータをするなです。

```
INSERT INTO `tableA` (`field_one`, `field_two`)
   SELECT `tableB`.`field_one`, `tableB`.`field_two`
   FROM `tableB`
   WHERE `tableB`.clmn <> 'someValue'
   ORDER BY `tableB`.`sorting_clmn`;
```

`SELECT * FROM`できますが、 `tableA`と`tableB`とするデータがしているがあります。

`AUTO_INCREMENT`は、 `INSERT` with `VALUES`のようにわれます。

このをすると、のテーブルのデータでにテーブルをめることができますさらに、にデータをフィルタリングする。

AUTO_INCREMENT + LAST_INSERT_IDによるINSERT

に`AUTO_INCREMENT PRIMARY KEY`がある、はそのにはされません。わりに、のをすべてしてから、しいIDがであるかをねます。

```
CREATE TABLE t (
    id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
    this ...,
    that ...,
    PRIMARY KEY(id) );

INSERT INTO t (this, that) VALUES (..., ...);
SELECT LAST_INSERT_ID() INTO @id;
INSERT INTO another_table (..., t_id, ...) VALUES (..., @id, ...);
```

`LAST_INSERT_ID()`はセッションにびついているので、のがじテーブルにされていても、それぞれがのIDをします。

あなたのクライアントAPIはおそらくに`SELECT`をせずにをクライアントにすことなく、`LAST_INSERT_ID()`をるのを`@variable`ています。これがましい。

よりく、よりな

IODKUの「の」は、`AUTO_INCREMENT PRIMARY KEY`ではなく、`UNIQUE`キーについて「キー」をトリガーすることです。はそのようなことをしています。これは、INSERTに`id`をしないことにしてください。

ののセットアップ

```
CREATE TABLE iodku (
    id INT AUTO_INCREMENT NOT NULL,
    name VARCHAR(99) NOT NULL,
    misc INT NOT NULL,
    PRIMARY KEY(id),
    UNIQUE(name)
) ENGINE=InnoDB;

INSERT INTO iodku (name, misc)
    VALUES
    ('Leslie', 123),
    ('Sally', 456);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  |  456 |
+----+--------+------+
```

IODKUが「」を`LAST_INSERT_ID()`、する`id`する`LAST_INSERT_ID()`

```
INSERT INTO iodku (name, misc)
    VALUES
    ('Sally', 3333)            -- should update
    ON DUPLICATE KEY UPDATE    -- `name` will trigger "duplicate key"
    id = LAST_INSERT_ID(id),
    misc = VALUES(misc);
SELECT LAST_INSERT_ID();       -- picking up existing value
+------------------+
```

```
| LAST_INSERT_ID() |
+------------------+
|                2 |
+------------------+
```

IODKUが「」をし、`LAST_INSERT_ID()`がしい`id`する

```
INSERT INTO iodku (name, misc)
    VALUES
    ('Dana', 789)          -- Should insert
    ON DUPLICATE KEY UPDATE
    id = LAST_INSERT_ID(id),
    misc = VALUES(misc);
SELECT LAST_INSERT_ID();   -- picking up new value
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                3 |
+------------------+
```

のテーブルの

```
SELECT * FROM iodku;
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  | 3333 |  -- IODKU changed this
|  3 | Dana   |  789 |  -- IODKU added this
+----+--------+------+
```

われたAUTO_INCREMENT ids

いくつかの"でIDをくことができます。は、InnoDBをったですのエンジンはがなるかもしれません

```
CREATE TABLE Burn (
    id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
    name VARCHAR(99) NOT NULL,
    PRIMARY KEY(id),
    UNIQUE(name)
        ) ENGINE=InnoDB;

INSERT IGNORE INTO Burn (name) VALUES ('first'), ('second');
SELECT LAST_INSERT_ID();          -- 1
SELECT * FROM Burn ORDER BY id;
  +----+--------+
  |  1 | first  |
  |  2 | second |
  +----+--------+

INSERT IGNORE INTO Burn (name) VALUES ('second');  -- dup 'IGNOREd', but id=3 is burned
SELECT LAST_INSERT_ID();          -- Still "1" -- can't trust in this situation
SELECT * FROM Burn ORDER BY id;
  +----+--------+
```

```
  |  1 | first  |
  |  2 | second |
  +----+--------+

INSERT IGNORE INTO Burn (name) VALUES ('third');
SELECT LAST_INSERT_ID();           -- now "4"
SELECT * FROM Burn ORDER BY id;    -- note that id=3 was skipped over
  +----+--------+
  |  1 | first  |
  |  2 | second |
  |  4 | third  |    -- notice that id=3 has been 'burned'
  +----+--------+
```

それをえるおおよそこのは、まず、インサートがされるがありますどのようにくのをすることになります。に、そのテーブルのauto_incrementからくのをします。に、にじてIDをしてをし、りのをく。

システムがシャットダウンしてした、っているをできるのはのです。すると、に`MAX(id)`がされます。これは、かれたID、またはもいidの`DELETE`sによってされたIDをすることがあります。

に、 `INSERT` `DELETE` + `INSERT`である`REPLACE`をむのフレーバーはIDをくことができます。 InnoDBでは、グローバルなセッションではない`innodb_autoinc_lock_mode`をって、がこっているのかをできます。

いを`AUTO_INCREMENT` `id`に「」すると、にきみがわれます。これは、のきされるにつながるが `INT`あなたがんだの。

オンラインでインサートをむ https://riptutorial.com/ja/mysql/topic/866/インサート

28: インデックスとキー

- - なインデックスをする

  CREATE INDEX *index_name* ON *table_name* *column_name1* [、 *column_name2* 、...]

- - ユニークなインデックスをする

  *table_name*でUNIQUE INDEXを*index_name*のを *CREATEcolumn_name1 [*、column_name2、...]

- - ドロップインデックス

  DROP INDEX **index_name** ON **tbl_name** [アルゴリズム _オプション | **lock_option**] ...

  algorithm_optionアルゴリズム [=] {DEFAULT | INPLACE | COPY}

  lock_option LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}

コンセプト

MySQLテーブルのインデックスはブックのインデックスのようにします。

たとえば、データベースにするがあり、ストレージなどのをしたいとします。がなければのようなのがないとして、トピックをつけられるまでそれは「スキャン」、ページを1つずつべなければなりません。、にはキーワードのリストがあるので、をして、そのが113-120,231、および354ページにされていることをします。に、せずにそれらのページにできますインデックスきの、ややい。

もちろん、インデックスのはくのにします - いくつかのでは、のsimileをします

- あなたがデータベースにするをっていて、 "データベース"というをけしているは、1-59ページ、61-290ページ、および292-400ページにされていることがあります。それはくのページであり、そのような、インデックスはそれほどつものではなく、ページをにべるほうがいかもしれません。 データベースでは、これは「のさ」です。
- 10ページのについては、5ページのインデックスがプレフィックスされた10ページのでわるがあるため、インデックスをするのはがありません。ちょうどかです。10ページをスキャンしてします。
- また、インデックスはであるがあります。たとえば、ページあたりの「L」のなど、にインデックスをするはありません。

Examples

インデックスをする

```
-- Create an index for column 'name' in table 'my_table'
CREATE INDEX idx_name ON my_table(name);
```

ユニークなインデックスをする

ユニークなインデックスは、したデータをテーブルにすることをぎます。のをするにNULLをすることができます、 NULLはのNULLをむのとはなるため

```
-- Creates a unique index for column 'name' in table 'my_table'
CREATE UNIQUE INDEX idx_name ON my_table(name);
```

ドロップインデックス

```
-- Drop an index for column 'name' in table 'my_table'
DROP INDEX idx_name ON my_table;
```

インデックスをする

これにより、のキー mystringおよびmydatetimeインデックスがされ、 WHEREののカラムでクエリがされます。

```
CREATE INDEX idx_mycol_myothercol ON my_table(mycol, myothercol)
```

はですクエリにWHEREののがまれていないは、ものインデックスのみをできます。この、クエリとmycolでWHERE 、インデックス、しクエリをしますmyothercolもしなし mycol ませんが。については、 このブログをください 。

BTREEののため、はでされるはものになります。たとえば、 DATETIMEカラムは、 WHERE datecol > '2016-01-01 00:00:00'ようにはされWHERE datecol > '2016-01-01 00:00:00' 。 BTREEインデックスはをににしますが、としてクエリされたがインデックスののインデックスであるにります。

AUTO_INCREMENTキー

```
CREATE TABLE (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    ...
    PRIMARY KEY(id),
    ...  );
```

メインノート

- 1からまり、 INSERTにしなかったはに1ずつインクリメントするか、 NULLとしてしNULL。
- IDはにいにされますが...

- のにではなく、idのについてのらかのギャップがなく、にされ、されないなどをしないでください。

なノート

- サーバーのに、 'のは`MAX(id)+1`として 'されます'。
- シャットダウンまたはクラッシュののがのIDをするは 、そのID をできますこれはエンジンにします。だから、 *auto_increments*がにユニークであるとしないでください 。らはいつでもでしかありません。
- マルチマスターソリューションまたはクラスタソリューションについては、`auto_increment_offset`および`auto_increment_increment`してください。
- `PRIMARY KEY`としてかのものをち、に`INDEX(id)`することはOKです。 これはによってはです。
- `AUTO_INCREMENT`を「 `PARTITION`キー」としてすることはめったにではありません。うことをする。
- さまざまなによってが「きけられる」ことがあります。 `INSERT IGNORE` dupキーを、 `REPLACE` `DELETE` + `INSERT` などをしないで、をりりするときにします。 `ROLLBACK`はIDののもうつのです。
- レプリケーションでは、IDがでスレーブにするのをできません。 IDはしたでりてられますが、InnoDBステートメントは`COMMIT`にスレーブにされます。

オンラインでインデックスとキーをむ https://riptutorial.com/ja/mysql/topic/1748/インデックスとキー

29: エラー1055ONLY_FULL_GROUP_BYかがGROUP BYにない...

き

、しいバージョンのMySQLサーバでは、これまでいていたクエリにして1055のエラーがしめました。このトピックでは、これらのエラーについてします。 MySQLチームは、な`GROUP BY`を`GROUP BY`しようとしています。なくとも、クエリがそれをくのがしくなっています。

い、MySQLには`GROUP BY`にするながまれていました。これはのでなをにします。このにより、ののが、をしているのかをにすることなく、コードで`GROUP BY`をすることができました。

に、の`GROUP BY`でをするがあるため、 `GROUP BY`せで`SELECT *`をすることはましくありません。なことに、くのがこれをっています。

これをむ。 https://dev.mysql.com/doc/refman/5.7/en/group-by-handling.html

MySQLチームは、コードをすことなくこのったをしようとしています。らは`sql_mode`5.7.5にフラグを`ONLY_FULL_GROUP_BY`のをします。のリリースでは、デフォルトでフラグをオンにしていました。ローカルMySQLを5.7.14にアップグレードすると、フラグがオンになり、のにするプロダクションコードがしなくなりました。

1055のエラーがした、どのようなができますか

1. のSQLクエリをするか、にそのをえることができます。
2. しているアプリケーションソフトウエアとのあるMySQLのバージョンにロールバックしてください。
3. サーバーの`sql_mode`をして、しくされた`ONLY_FULL_GROUP_BY`モードを`ONLY_FULL_GROUP_BY`ます。

`SET`コマンドをすると、モードをできます。

```
SET  sql_mode =
'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENG
```

あなたのアプリケーションがMySQLにしたにそれをうなら、このトリックをうべきです。

それとも、あなたはつけることができますMySQLのインストールにファイルを 、つけ`sql_mode=`をし、し、それを`ONLY_FULL_GROUP_BY` 、そしてサーバーをしてください。

Examples

GROUP BYのと

```
SELECT item.item_id, item.name,     /* not SQL-92 */
       COUNT(*) number_of_uses
 FROM item
 JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

`item`とばれるテーブルにがされ、 `uses`というテーブルにするのがされ`uses`。これはにしますが、ながらのSQL-92ではありません。

なの `GROUP BY`せの`SELECT`および`ORDER BY`には、のようながまれているがあるためです。

1. `GROUP BY`にされている
2. `COUNT()` 、 `MIN()`などの。

こののSELECTには、いずれかのをたすではない`item.name`ています。 MySQL 5.6では、SQLモードに`ONLY_FULL_GROUP_BY`がまれている、このクエリはされ`ONLY_FULL_GROUP_BY`。

こののクエリは、このように`GROUP BY`をすることによってSQL-92にするようにすることができます。

```
SELECT item.item_id, item.name,
       COUNT(*) number_of_uses
  FROM item
  JOIN uses ON item.item_id, uses.item_id
 GROUP BY item.item_id, item.name
```

のSQL-99では、DBMSがグループキーとののをできる、 `SELECT`でされていないをグループキーからすることができます。ので`item.name`のにしている`item.item_id` 、のでは、なSQL-99です。MySQLはバージョン5.7でプローバをしました。のは、 `ONLY_FULL_GROUP_BY`で`ONLY_FULL_GROUP_BY`ます。

GROUP BYをってできないをすMurphy's Law

```
SELECT item.item_id, uses.category,   /* nonstandard */
       COUNT(*) number_of_uses
  FROM item
  JOIN uses ON item.item_id, uses.item_id
 GROUP BY item.item_id
```

itemとばれるテーブルにがされ、usesというテーブルにするのがされます。また、 `uses.category`というのもされ`uses.category`。

このクエリは、 `ONLY_FULL_GROUP_BY`フラグがれるにMySQLでします。これは、 MySQLのなを`GROUP BY`にします 。

しかし、せにはがあります。 `uses`ののがJOINのONとすると、MySQLはそのの1つから`category`をします。どのクエリのとアプリケーションのユーザーは、にそのことをることはできません。にえば、 できないことです.MySQLはなをすことができます。

とは、 ランダムなものとですが、きないが1つあります。 ランダムなが々わるとされるかもしれません。したがって、がランダムであった、デバッグまたはテストにそのをするがあります。 しないがよりします。MySQLはクエリをするたびにじをします。には、それはのをきこすMySQLサーバのしいバージョンです。ときにはをきこすテーブルがえています。ってくことができます、ってくと、あなたがそれをしていないとき。それはマーフィーのです。

MySQLチームは、がこのいをすのをよりにするようめてきました。 5.7シーケンスのしいバージョンのMySQLには、 `sql_mode`という`sql_mode`フラグが`ONLY_FULL_GROUP_BY`ます。このフラグがされると、MySQLサーバは1055エラーをし、こののクエリのをします。

SELECT *をしてGROUP BYをし、それをする。

には、 `SELECT`に`*`をけてすると、このようになります。

```
SELECT item.*,      /* nonstandard */
       COUNT(*) number_of_uses
 FROM item
 JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

このようなクエリは、 `ONLY_FULL_GROUP_BY`にするようにリファクタリングするがあります。

これをうには、 `item_id number_of_uses`をしくすために`GROUP BY`しくするサブクエリが`item_id`。 このサブクエリは、 `uses`テーブルをべるだけでみ、くていです。

```
                            SELECT item_id, COUNT(*) number_of_uses
                              FROM  uses
                             GROUP BY item_id
```

に、そのサブクエリを`item`テーブルとすることができます。

```
SELECT item.*, usecount.number_of_uses
  FROM item
  JOIN (
                            SELECT item_id, COUNT(*) number_of_uses
                              FROM  uses
                             GROUP BY item_id
       ) usecount ON item.item_id = usecount.item_id
```

これにより、 `GROUP BY`をでにすることができます。また、 `*`をすることもできます。

それにもかかわらず、なはいずれのでも`*`のをけます。クエリになをするがです。

ANY_VALUE

```
SELECT item.item_id, ANY_VALUE(uses.tag) tag,
       COUNT(*) number_of_uses
 FROM item
 JOIN uses ON item.item_id, uses.item_id
```

```
GROUP BY item.item_id
```

itemというテーブルの、するの、およびusesというテーブルののの1つをします。

このANY_VALUE()は、なのとえることができます。 count、sum、またはmaximumをすわりに、MySQLサーバに、のグループから1つのをにするようにします。エラー1055をするです。

プロダクションアプリケーションのクエリでANY_VALUE()をするはがです。

それはにSURPRISE_ME()とばれるべきです。 GROUP BYグループのののをします。それがすはです。つまり、それはMySQLサーバーまでです。には、なをします。

サーバーはランダムなをしません。それよりもいです。クエリがされるたびにじがされます。テーブルがきくなったりさくなったり、サーバのRAMがかったり、サーバのバージョンがわったり、ににかがあったとしても、まったくなしにすることができます。

あなたはされています。

オンラインでエラー1055ONLY_FULL_GROUP_BYかがGROUP BYにない...をむ
https://riptutorial.com/ja/mysql/topic/8245/エラー1055-only-full-group-by-かがgroup-byにない---

30: エラーコード

Examples

エラーコード1064エラー

```
select LastName, FirstName,
from Person
```

メッセージ

> エラーコード1064. SQLにエラーがあります。しいについてはMySQLサーバのバージョンにするマニュアルをチェックし、2の 'from Person'のくでしてください。

MySQLから「1064エラー」メッセージをけると、エラーなしではクエリをできません。いえれば、クエリのをできません。

エラーメッセージのは、MySQLがするをできないクエリののでまります。このでは、MySQLはで`from Person`をなさない。この、 `from Person`になカンマがあります。コンマは、 `SELECT`にののがあるとMySQLにします

エラーは、に`... near '...'`ます。のにあるものは、エラーがあるににいところにあります。エラーをつけるには、ののトークンとののののトークンをてください。

々、あなたは`... near ''`るでしょう。つまり、ではもされません。つまり、MySQLができないのは、ステートメントのまたはにてはまります。これは、クエリがアンバランスがまれているして'または"またはアンバランスまたはあなたがにしくをしなかったこと。

ストアドルーチンの、 `DELIMITER`をしくするのをれているかもしれません。

したがって、エラー1064がしたは、クエリのテキストをて、エラーメッセージにされているをつけます。そのをとしたクエリのテキストをにべます。

かにエラー1064のトラブルシューティングをするは、クエリのテキストとエラーメッセージのテキストのをすることをおめします。

エラーコード1175セーフアップデート

このエラーは、 `KEY`をする`WHERE`をめずにレコードをまたはしようとしているときにされます。

またはをするには、のようにします。

```
SET SQL_SAFE_UPDATES = 0;
```

セーフモードをにするには、のようにします。

```
SET SQL_SAFE_UPDATES = 1;
```

エラーコード1215キーをできません。

このエラーは、がけているキー `FK` のなルックアップをするために、テーブルがにされていないにします。

```
CREATE TABLE `gtType` (
  `type` char(2) NOT NULL,
  `description` varchar(1000) NOT NULL,
  PRIMARY KEY (`type`)
) ENGINE=InnoDB;

CREATE TABLE `getTogethers` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` char(2) NOT NULL,
  `eventDT` datetime NOT NULL,
  `location` varchar(1000) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_gt2type` (`type`), -- see Note1 below
  CONSTRAINT `gettogethers_ibfk_1` FOREIGN KEY (`type`) REFERENCES `gtType` (`type`)
) ENGINE=InnoDB;
```

1このようなKEYは、それにくのFKのためににじてにされます。はスキップすることができ、にじてKEYインデックスがされます。がスキップしているを`someOther`します。

これまでのところとてもい、のびしまで。

```
CREATE TABLE `someOther` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `someDT` datetime NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `someOther_dt` FOREIGN KEY (`someDT`) REFERENCES `getTogethers` (`eventDT`)
) ENGINE=InnoDB;
```

> エラーコード1215.キーをできません

この、 `eventDT`をするために、 されるテーブル`getTogethers`にインデックスがないためにします。のでされる。

```
CREATE INDEX `gt_eventdt` ON getTogethers (`eventDT`);
```

テーブル`getTogethers`がされ、 `someOther`のがするようになりました。

FOREIGN KEYをしたMySQLマニュアルページから

> MySQLでは、キーのチェックがでテーブルスキャンをとしないように、キーとキーのインデックスがです。では、キーがじでのとしてリストされるがです。このようなは、にしないはにされます。
>
> キーとキーのするには、のデータがです。のサイズとはじでなければなりません。の

さはじであるはありません。バイナリの、セットとはじでなければなりません。

InnoDBは、キーがインデックスカラムまたはカラムのグループをすることをします。ただし、のには、のがじでのとしてリストされるがです。

ののののの、およびキーのいにアドバイスされていますにしてください。

テーブルがにされると、にされたキーは、のようなコマンドでされます。

```
SHOW CREATE TABLE someOther;
```

このエラーをするのなケースには、のようにドキュメントがまれますが、するがあります。

- `INT UNSIGNED`してされた`INT`かけのない。
- マルチカラムキーおよびのもののにがある。

1045アクセスがされました

「GRANT」と「rootパスワードの」のをしてください。

1236「な」

これはマスターがクラッシュし、 `sync_binlog`がオフであったことをします 。は、スレーブののbinlogファイルマスターをの`CHANGE MASTER to POS=0`に`CHANGE MASTER to POS=0`することです。

マスタは、 `sync_binlog=OFF`そのバイナリログにフラッシュするにスレーブにレプリケーションアイテムをします。フラッシュのにマスタがクラッシュした、スレーブはすでににバイナリログのファイルのをしています。マスタがびすると、しいバイナリログがされるので、そのバイナリログのにすることがのです。

なI / Oがするがある、なは`sync_binlog=ON`です。

GTIDをしているは...

2002、2003できません

ファイアウォールのをブロックするポート3306をします。

いくつかのなおよび/またはは

- サーバーはにしていますか
- "service firewalld stop"および "systemctl disable firewalld"
- Telnetマスター3306
- `bind-address`する
- check `skip-name-resolve`
- ソケットをしてください。

1067、1292、1366、1411 - 、、、デフォルトなどのが

1067これはおそらく`TIMESTAMP`デフォルトにしており、がつにつれてしています。 DatesTimesペ―ジの`TIMESTAMP defaults`をしてください。 まだしない

1292/1366 DOUBLE / Integerやそののエラ―をチェックします。がっていることをします。おそらく、あなたは`VARCHAR`れているとうかもしれませんが、それはのにえられています。

1292 DATETIMEまたはがすぎるかどうかをチェックします。がされた2から3までをチェックしてください。 `+00`タイムゾ―ンのようななながないかしてください。

1292 VARIABLE `SET`しようとしている`VARIABLE`をしてください。

1292 LOAD DATA「い」をてください。エスケ―プなどをします。デ―タをします。

1411 STR_TO_DATEフォ―マットされたがっていますか

126,127,134,144,145

MySQLデ―タベ―スからレコ―ドにアクセスしようとすると、これらのエラ―メッセ―ジがされることがあります。これらのエラ―メッセ―ジは、MySQLデ―タベ―スののためにしました。はタイプです

```
MySQL error code 126 = Index file is crashed
MySQL error code 127 = Record-file is crashed
MySQL error code 134 = Record was already deleted (or record file crashed)
MySQL error code 144 = Table is crashed and last repair failed
MySQL error code 145 = Table was marked as crashed and should be repaired
```

MySQLのバグ、ウイルス、サ―バ―クラッシュ、なシャットダウン、したテ―ブルがこののです。それがれてしまうと、アクセスになり、もはやアクセスできなくなります。アクセシビリティをるには、されたバックアップからデ―タをするのです。ただし、されたバックアップやなバックアップがないは、MySQLのにむことができます。

テ―ブルエンジンタイプが`MyISAM`は、 `CHECK TABLE`し、に`REPAIR TABLE`をします。

InnoDBへのについてにえてください。このエラ―はびこりません。

```
CHECK TABLE <table name> ////To check the extent of database corruption
REPAIR TABLE <table name> ////To repair table
```

139

エラ―139は、テ―ブルのフィ―ルドのとサイズがあるをえていることをするがあります。

- スキ―マをする
- いくつかのフィ―ルドをする

- テーブルをにする

1366

これは、、セットのがクライアントとサーバーのでしていないことをします。はこちらをごください。

126、1054、1146、1062、24

をるこれらの4つのエラーをめると、このページではユーザーのなエラーの50をカバーするといます。

はい、この「」にはリビジョンがです。

24ファイルをくことができませんいているファイルがすぎます

`open_files_limit`はOSのにします。 `table_open_cache`はそれより`table_open_cache`するがあります。

これらのエラーがするがあります

- ストアドプロシージャの`DEALLOCATE PREPARE`にしました。
- のパーティションとinnodb_file_per_table = ONのPARTITIONEDテーブル。えられたテーブルに50のパーティションをたないようにしてくださいさまざまなで。「ネイティブパーティション」がになると、このアドバイスはされるがあります。

らかには、OSのをやすことですよりくのファイルをするには、 `ulimit`または`/etc/security/limits.conf`するか、 `sysctl.conf` kern.maxfileskern.maxfilesperprocまたはのものOSにをします。に、 `open_files_limit`と`table_open_cache`ます。

5.6.8、 `open_files_limit`は`max_connections`についてサイズされていますが、デフォルトからすることはです。

1062 - エントリ

このエラーは、にの2つのによりします

1. - `Error Code: 1062. Duplicate entry '12' for key 'PRIMARY'`

   キーはであり、エントリをけれません。したがって、すでにするしいをしようとすると、このエラーがします。

   これをするには、キーを`AUTO_INCREMENT`にします。また、しいをしようとするときは、キーをするか、 `NULL`をキーにし`NULL`。

```
CREATE TABLE userDetails(
  userId INT(10) NOT NULL AUTO_INCREMENT,
  firstName VARCHAR(50),
```

```
  lastName VARCHAR(50),
  isActive INT(1) DEFAULT 0,
  PRIMARY KEY (userId) );

--->and now while inserting
INSERT INTO userDetails VALUES (NULL ,'John', 'Doe', 1);
```

2. ユニークなデータフィールド - `Error Code: 1062. Duplicate entry 'A' for key 'code'`

   のをりてて、そのににするをつしいをしようとすると、このエラーがします。

   このエラーをするには、の`INSERT`ではなく`INSERT IGNORE`をし`INSERT IGNORE`。しようとしているしいがのレコードをしていない、MySQLはそれをりします。レコードがしている、`IGNORE`キーワードはエラーをせずにします。

```
INSERT IGNORE INTO userDetails VALUES (NULL ,'John', 'Doe', 1);
```

オンラインでエラーコードをむ https://riptutorial.com/ja/mysql/topic/895/エラーコード

31: キャラクタセットと

Examples

```
CREATE TABLE foo ( ...
    name CHARACTER SET utf8mb4
    ... );
```

セットをするためには、クライアントのバイトがどのようなエンコーディングであるかをMySQLサーバにえることがです。これはつのです

```
SET NAMES utf8mb4;
```

PHP、Python、Java、...にはのがありますが、は`SET NAMES`よりもましいです。

えば `SET NAMES utf8mb4` 、とに`CHARACTER SET latin1` -これは、ときutf8mb4するlatin1のからします`INSERTing`し、るときに`SELECTing`。

どのキャラクターセットとコレクションですか

ものをむのキャラクタセットがあります。 1つのは1つのセットにのみします `SHOW COLLATION;`をしてください`SHOW COLLATION;`。

は4つの`CHARACTER SETs`のみがです。

```
ascii -- basic 7-bit codes.
latin1 -- ascii, plus most characters needed for Western European languages.
utf8 -- the 1-, 2-, and 3-byte subset of utf8.  This excludes Emoji and some of Chinese.
utf8mb4 -- the full set of UTF8 characters, covering all current languages.
```

すべてのがまれ、じようにエンコードされます。 utf8はutf8mb4のサブセットです。

ベストプラクティス...

- utf8mb4は、さまざまなをできる`TEXT`または`VARCHAR`にします。
- 16UUID、MD5などとシンプルコードcountry_code、postal_codeなどにはasciilatin1はOKをします。

utf8mb4はバージョン5.5.3までしなかったので、utf8はそれになのものでした。

*MySQL*のでは 、 "UTF8"は、MySQLのutf8mb4ではなく、MySQLのutf8mb4とじものをします。

はセットでまり、は "case and accent insensitive"のは`_bin` 、 "にビットをする"は`_ci`でわります。

「の」utf8mb4は、Unicode 5.20につく`utf8mb4_unicode_520_ci`です。 1つのでしているは、ポーラ

ンドのについてをしべえる`utf8mb4_polish_ci`がきかもしれません。

とフィールドのセットの

セットは、 `CHARACTER SET`と`CHARSET`をして、テーブルごと、および々のフィールドごとにできます。

```
CREATE TABLE Address (
    `AddressID`    INTEGER NOT NULL PRIMARY KEY,
    `Street`       VARCHAR(80) CHARACTER SET ASCII,
    `City`         VARCHAR(80),
    `Country`      VARCHAR(80) DEFAULT "United States",
    `Active`       BOOLEAN DEFAULT 1,
) Engine=InnoDB default charset=UTF8;
```

`City`と`Country`は`UTF8`をします。これは、これをテーブルのデフォルトのセットとしてします。、`Street`は`ASCII`をします。にはそうしています。

しいセットをすることは、データセットにきくしますが、データをうシステムのをにさせることもできます。

オンラインでキャラクタセットとをむ https://riptutorial.com/ja/mysql/topic/4569/キャラクタセットと

32: クラスタリング

Examples

さ

"MySQL Cluster"のさ

- NDBクラスタ - された、にメモリのエンジン。くわれていません。
- GaleraクラスタPercona XtraDBクラスタ、PXC、GaleraとMariaDB。 - MySQLのにいハイアベイラビリティソリューション。レプリケーションをえています。

これらの "クラスタ"のページをしてください。

「クラスタインデックス」については、 `PRIMARY KEY`のページをしてください。

オンラインでクラスタリングをむ https://riptutorial.com/ja/mysql/topic/5130/クラスタリング

33: グループする

1. SELECT expression1、expression2、... expression_n、
2. aggregate_functionexpression
3. FROMテーブル
4. [WHERE]
5. GROUP BY expression1、expression2、... expression_n;

パラメーター

パラメータ	
1、2、... _n	にカプセルされていないで、GROUP BYにめるがあります。
	SUM、COUNT、MIN、MAX、またはAVGなどの。
テーブル	あなたがレコードをしたいテーブル。 FROMにはなくとも1つのテーブルがリストされているがあります。
WHERE	オプション。レコードをするためにたすがある。

MySQL GROUP BYは、SELECTでのレコードでデータをし、を1つのでグループするためにされます。

そのるいは、 `ONLY_FULL_GROUP_BY`のによってにされます 。これをにすると、にまれないでグループされた`SELECT`がエラーをします。 これは5.7.5のデフォルトです 。このをしたりしたりしないと、のDBMSにれしんだユーザーやユーザーにとってがするがあります。

Examples

GROUP BY USING SUM

```
SELECT product, SUM(quantity) AS "Total quantity"
FROM order_details
GROUP BY product;
```

MINをしてグループする

が`name`、 `department`、 `salary`をつであるのテーブルをします。

```
SELECT department, MIN(salary) AS "Lowest salary"
```

```
FROM employees
GROUP BY department;
```

これは、どのにのがまれているか、そのはかをします。でののnameをつけることは、こののののです。「groupwise max」をしてください。

GROUP BY USING COUNT

```
SELECT department, COUNT(*) AS "Man_Power"
FROM employees
GROUP BY department;
```

HAVINGをしてGROUP BY

```
SELECT department, COUNT(*) AS "Man_Power"
FROM employees
GROUP BY department
HAVING COUNT(*) >= 10;
```

`GROUP BY ... HAVING`をするレコードをフィルタリングすることは、 `SELECT ... WHERE`をして々のレコードをフィルタリングすることにています。

`HAVING`が"エイリアス"をしているので、 `HAVING HAVING Man_Power >= 10`とうこともできます。

Group Concatをしてグループする

Group Concatは、MySQLでされ、ごとにのをつのをします。つまり、 `Name(1):Score(*)`などの1つのにしてされるがあります`Name(1):Score(*)`

	スコア
アダム	A +
アダム	A-
アダム	B
アダム	C +
ビル	D-
ジョン	A-

```
SELECT Name, GROUP_CONCAT(Score ORDER BY Score desc SEPERATOR ' ') AS Grades
FROM   Grade
GROUP BY Name
```

```
+------+------------+
| Name | Grades     |
+------+------------+
| Adam | C+ B A- A+ |
| Bill | D-         |
| John | A-         |
+------+------------+
```

AGGREGATEをつGROUP BY

テーブルオーダー

```
+---------+------------+----------+-------+--------+
| orderid | customerid | customer | total | items  |
+---------+------------+----------+-------+--------+
|       1 |          1 | Bob      |  1300 |     10 |
|       2 |          3 | Fred     |   500 |      2 |
|       3 |          5 | Tess     |  2500 |      8 |
|       4 |          1 | Bob      |   300 |      6 |
|       5 |          2 | Carly    |   800 |      3 |
|       6 |          2 | Carly    |  1000 |     12 |
|       7 |          3 | Fred     |   100 |      1 |
|       8 |          5 | Tess     | 11500 |     50 |
|       9 |          4 | Jenny    |   200 |      2 |
|      10 |          1 | Bob      |   500 |     15 |
+---------+------------+----------+-------+--------+
```

- カウント

WHEREでのをたすをします。

の。

```
SELECT customer, COUNT(*) as orders
FROM orders
GROUP BY customer
ORDER BY customer
```

```
+----------+--------+
| customer | orders |
+----------+--------+
| Bob      |      3 |
| Carly    |      2 |
| Fred     |      2 |
| Jenny    |      1 |
| Tess     |      2 |
+----------+--------+
```

-

したのをします。

とのアイテムの。

```
SELECT customer, SUM(total) as sum_total, SUM(items) as sum_items
FROM orders
GROUP BY customer
ORDER BY customer
```

```
+----------+-----------+-----------+
| customer | sum_total | sum_items |
+----------+-----------+-----------+
| Bob      |      2100 |        31 |
| Carly    |      1800 |        15 |
| Fred     |       600 |         3 |
| Jenny    |       200 |         2 |
| Tess     |     14000 |        58 |
+----------+-----------+-----------+
```

- **AVG**

ののをします。

の

```
SELECT customer, AVG(total) as avg_total
FROM orders
GROUP BY customer
ORDER BY customer
```

```
+----------+-----------+
| customer | avg_total |
+----------+-----------+
| Bob      |       700 |
| Carly    |       900 |
| Fred     |       300 |
| Jenny    |       200 |
| Tess     |      7000 |
+----------+-----------+
```

- **MAX**

のまたはのをします。

の。

```
SELECT customer, MAX(total) as max_total
FROM orders
GROUP BY customer
ORDER BY customer
```

```
+----------+-----------+
| customer | max_total |
+----------+-----------+
| Bob      |      1300 |
| Carly    |      1000 |
| Fred     |       500 |
| Jenny    |       200 |
```

```
| Tess     |     11500 |
+----------+-----------+
```

- **MIN**

のまたはのをします。

の。

```
SELECT customer, MIN(total) as min_total
FROM orders
GROUP BY customer
ORDER BY customer
```

```
+----------+-----------+
| customer | min_total |
+----------+-----------+
| Bob      |       300 |
| Carly    |       800 |
| Fred     |       100 |
| Jenny    |       200 |
| Tess     |      2500 |
+----------+-----------+
```

オンラインでグループするをむ https://riptutorial.com/ja/mysql/topic/3523/グループする

34: コメントMysql

のスペースをとする--スタイルのコメントは、スペースをとしないSQLとはがなります 。

Examples

コメントの

コメントには3つのタイプがあります

```
 # This comment continues to the end of line

-- This comment continues to the end of line

/* This is an in-line comment */

/*
This is a
multiple-line comment
*/
```

```
SELECT * FROM t1; -- this is comment

CREATE TABLE stack(
    /*id_user int,
    username varchar(30),
    password varchar(30)
    */
    id int
);
```

--このは、スペースは、のことがです--コメントがまるに、それのは、コマンドとしてされ、はエラーがします。

```
#This comment works
/*This comment works.*/
--This comment does not.
```

テーブルのコメント

```
CREATE TABLE menagerie.bird (
    bird_id INT NOT NULL AUTO_INCREMENT,
    species VARCHAR(300) DEFAULT NULL COMMENT 'You can include genus, but never subspecies.',
    INDEX idx_species (species) COMMENT 'We must search on species often.',
    PRIMARY KEY (bird_id)
) ENGINE=InnoDB COMMENT 'This table was inaugurated on February 10th.';
```

COMMENTの=はオプションです。 ドキュメント

これらのコメントは、となり、スキーマにされ、 `SHOW CREATE TABLE`または`information_schema`からできます。

オンラインでコメントMysqlをむ https://riptutorial.com/ja/mysql/topic/2337/コメントmysql

35: サーバー

パラメーター

パラメーター	
グローバル	サーバーにしてされているをします。オプション。
セッション	このセッションにのみされているをします。オプション。

Examples

SHOW VARIABLESの

すべてのサーバをするには、このクエリをインタフェースPHPMyAdminなどのSQLウィンドウまたはMySQL CLIインタフェース

```
SHOW VARIABLES;
```

セッションまたはグローバルをのようにすることができます。

セッション

```
SHOW SESSION VARIABLES;
```

グローバル

```
SHOW GLOBAL VARIABLES;
```

のSQLコマンドとに、LIKEコマンドなどのクエリにパラメータをできます。

```
SHOW [GLOBAL | SESSION] VARIABLES LIKE 'max_join_size';
```

または、ワイルドカードをする

```
SHOW [GLOBAL | SESSION] VARIABLES LIKE '%size%';
```

のように、WHEREパラメータをしてSHOWクエリのをフィルタリングすることもできます。

```
SHOW [GLOBAL | SESSION] VARIABLES WHERE VALUE > 0;
```

SHOW STATUSの

データベースサーバのステータスをするには、するインタフェースPHPMyAdminまたはそののSQLウィンドウまたはMySQL CLIインタフェースのいずれかでこのクエリをします。

```
SHOW STATUS;
```

あなたは、あなたのサーバーのSESSIONまたはGLOBALステータスをのようにけるかどうかをすることができますSession status

```
SHOW SESSION STATUS;
```

グローバルステータス

```
SHOW GLOBAL STATUS;
```

のSQLコマンドとに、LIKEコマンドなどのクエリにパラメータをできます。

```
SHOW [GLOBAL | SESSION] STATUS LIKE 'Key%';
```

またはWhereコマンド

```
SHOW [GLOBAL | SESSION] STATUS WHERE VALUE > 0;
```

GLOBALとSESSIONのないは、GLOBALをすると、サーバーとそのすべてのにするがされ、SESSIONはののののみがされることです。

オンラインでサーバーをむ https://riptutorial.com/ja/mysql/topic/9924/サーバー

36: さまざまなプログラミングをしたUTF-8との

。

Examples

Python

ソースコードの1または2コードのリテラルをutf8でエンコードする

```
# -*- coding: utf-8 -*-
```

```
db = MySQLdb.connect(host=DB_HOST, user=DB_USER, passwd=DB_PASS, db=DB_NAME,
        charset="utf8mb4", use_unicode=True)
```

Webページの、のいずれか

```
<meta charset="utf-8" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

PHP

php.iniこれはPHP 5.6のデフォルトです

```
default_charset UTF-8
```

ウェブページをするとき

```
header('Content-type: text/plain; charset=UTF-8');
```

MySQLにするとき

```
(for mysql:)   Do not use the mysql_* API!
(for mysqli:)  $mysqli_obj->set_charset('utf8mb4');
(for PDO:)     $db = new PDO('dblib:host=host;dbname=db;charset=utf8', $user, $pwd);
```

コードでは、ルーチンをしないでください。

データの、

```
<form accept-charset="UTF-8">
```

JSONの、 `\uxxxx`をけるために

```
$t = json_encode($s, JSON_UNESCAPED_UNICODE);
```

オンラインでさまざまなプログラミングをしたUTF-8との。をむ
https://riptutorial.com/ja/mysql/topic/7332/さまざまなプログラミングをしたutf-8との-

37: ストアドルーチンきおよび

パラメーター

パラメータ	
り	からされるデータをします。
り	RETURNののまたはは、のびしにされます。

ストアド·ルーチンは、プロシージャーまたはのいずれかです。

プロシージャーはCALLステートメントをしてびされ、をしてのみすことができます。

はのとにのからびすことができ、スカラをすことができます。

Examples

をする

のなは、にINT12します。

```
DELIMITER ||
CREATE FUNCTION functionname()
RETURNS INT
BEGIN
    RETURN 12;
END;
||
DELIMITER ;
```

のは、り DELIMITER || をどのようにするかをします;デフォルトのままにしておくと、がされるにするがあります;そのの;のにあるものはCREATEのわりとみなされますが、これははもまれません。

CREATE FUNCTIONをした、デリミタをデフォルトのにすがあります;ののコード DELIMITER ; のにされています。

こののはのとおりです。

```
SELECT functionname();
+----------------+
| functionname() |
+----------------+
|             12 |
+----------------+
```

もうしなしかしそれでもやっかいなは、パラメ―タをとり、それにをします

```
DELIMITER $$
CREATE FUNCTION add_2 ( my_arg INT )
  RETURNS INT
BEGIN
  RETURN (my_arg + 2);
END;
$$
DELIMITER ;

SELECT add_2(12);
+-----------+
| add_2(12) |
+-----------+
|        14 |
+-----------+
```

DELIMITERディレクティブとはなるをすることにしてください。 CREATEのにはれないをにすることができますが、は\\ 、 ||ようなの2をします。または$$ 。

、プロシ―ジャまたはトリガのまたはのにパラメ―タをにすることは、コマンドラインからクエリをするときににりをするがある、のGUIではりをするがないため、よいです。

されたをむプロシ―ジャの

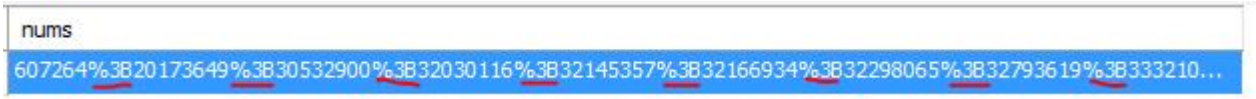
```
DROP PROCEDURE if exists displayNext100WithName;
DELIMITER $$
CREATE PROCEDURE displayNext100WithName
(    nStart int,
    tblName varchar(100)
)
BEGIN
    DECLARE thesql varchar(500); -- holds the constructed sql string to execute

    -- expands the sizing of the output buffer to accomodate the output (Max value is at least
4GB)
    SET session group_concat_max_len = 4096; -- prevents group_concat from barfing with error
1160 or whatever it is

    SET @thesql=CONCAT("select group_concat(qid order by qid SEPARATOR '%3B') as nums ","from
(    select qid from ");
    SET @thesql=CONCAT(@thesql,tblName," where qid>? order by qid limit 100 )xDerived");
    PREPARE stmt1 FROM @thesql; -- create a statement object from the construct sql string to
execute
    SET @p1 = nStart; -- transfers parameter passed into a User Variable compatible with the
below EXECUTE
    EXECUTE stmt1 USING @p1;

    DEALLOCATE PREPARE stmt1; -- deallocate the statement object when finished
END$$
DELIMITER ;
```

ストアドプロシ―ジャをすると、くのクライアントツ―ルでなDELIMITERがラップされます。

びし

```
call displayNext100WithName(1,"questions_mysql");
```

%3B セミコロンセパレータきサンプル



IN、OUT、INOUTパラメータをむストアドプロシージャ

```
DELIMITER $$

DROP PROCEDURE IF EXISTS sp_nested_loop$$
CREATE PROCEDURE sp_nested_loop(IN i INT, IN j INT, OUT x INT, OUT y INT, INOUT z INT)
BEGIN
    DECLARE a INTEGER DEFAULT 0;
    DECLARE b INTEGER DEFAULT 0;
    DECLARE c INTEGER DEFAULT 0;
    WHILE a < i DO
        WHILE b < j DO
            SET c = c + 1;
            SET b = b + 1;
        END WHILE;
        SET a = a + 1;
        SET b = 0;
    END WHILE;
    SET x = a, y = c;
    SET z = x + y + z;
END $$
DELIMITER ;
```

ストアドプロシージャをびす CALL

```
SET @z = 30;
call sp_nested_loop(10, 20, @x, @y, @z);
SELECT @x, @y, @z;
```

```
+------+------+------+
|  @x  |  @y  |  @z  |
+------+------+------+
|  10  |  200 |  240 |
+------+------+------+
```

INパラメータは、をプロシージャにします。プロシージャはをするがありますが、プロシージャがるときにははびしにはされません。

OUTパラメータは、プロシージャからのをびしにします。そのはプロシージャではNULLであり、そのはプロシージャがるときにびしにされます。

INOUTパラメーターは、びしによってされ、プロシージャーによってされ、プロシージャーによってわれたは、プロシージャーがったときにびしにされます。

Ref http : //dev.mysql.com/doc/refman/5.7/en/create-procedure.html

カーソル

カーソルをすると、クエリのを1ごとにりしできます。 `DECLARE`コマンドは、カーソルをし、のSQLクエリにけるためにされます。

```
DECLARE student CURSOR FOR SELECT name FROM studend;
```

いくつかののをしています。タイプのがいくつあるかをカウントしたいとえています。

たちのデータ

```
CREATE TABLE product
(
  id   INT(10) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  type VARCHAR(50)      NOT NULL,
  name VARCHAR(255)     NOT NULL

);
CREATE TABLE product_type
(
  name VARCHAR(50) NOT NULL PRIMARY KEY
);
CREATE TABLE product_type_count
(
  type  VARCHAR(50)      NOT NULL PRIMARY KEY,
  count INT(10) UNSIGNED NOT NULL DEFAULT 0
);

INSERT INTO product_type (name) VALUES
  ('dress'),
  ('food');

INSERT INTO product (type, name) VALUES
  ('dress', 'T-shirt'),
  ('dress', 'Trousers'),
  ('food', 'Apple'),
  ('food', 'Tomatoes'),
  ('food', 'Meat');
```

カーソルをしてストアドプロシージャをしてをすることができます。

```
DELIMITER //
DROP PROCEDURE IF EXISTS product_count;
CREATE PROCEDURE product_count()
  BEGIN
    DECLARE p_type VARCHAR(255);
    DECLARE p_count INT(10) UNSIGNED;
    DECLARE done INT DEFAULT 0;
    DECLARE product CURSOR FOR
      SELECT
        type,
        COUNT(*)
      FROM product
      GROUP BY type;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;
```

```
    TRUNCATE product_type;

    OPEN product;

    REPEAT
      FETCH product
      INTO p_type, p_count;
      IF NOT done
      THEN
        INSERT INTO product_type_count
        SET
          type  = p_type,
          count = p_count;
      END IF;
    UNTIL done
    END REPEAT;

    CLOSE product;
  END //
DELIMITER ;
```

プロシ―ジャをびすことができます

```
CALL product_count();
```

は`product_type_count`テ―ブルにあります

```
type   | count
----------------
dress  |   2
food   |   3
```

これは`CURSOR`いですが、プロシ―ジャ―をどのようにきえることができるのかをしてください

```
INSERT INTO product_type_count
        (type, count)
    SELECT type, COUNT(*)
        FROM product
        GROUP BY type;
```

これははるかにくされます。

のセット

`SELECT`ステ―トメントとはなり、 `Stored Procedure`はのセットをします。 Perl、PHPなどで`CALL`のをするためにされるなるコ―ドがです。

ここやのでのコ―ドがです

をする

```
DELIMITER $$
```

```
CREATE
    DEFINER=`db_username`@`hostname_or_IP`
    FUNCTION `function_name`(optional_param data_type(length_if_applicable))
    RETURNS data_type
BEGIN
    /*
    SQL Statements goes here
    */
END$$
DELIMITER ;
```

RETURNS data_typeはのMySQLデータです。

オンラインでストアドルーチンきおよびをむ https://riptutorial.com/ja/mysql/topic/1351/ストアドルーチン-きおよび-

38: スパースまたはデータの

Examples

NULLをむの

MySQLなどのSQLでは、 `NULL`にはなプロパティがあります。

、らがいていた、およびらがしたをむのをえてみましょう。 `NULL`は、がまだでいていることをし `NULL`。

```
CREATE TABLE example
(`applicant_id` INT, `company_name` VARCHAR(255), `end_date` DATE);

+--------------+-----------------+------------+
| applicant_id | company_name    | end_date   |
+--------------+-----------------+------------+
|            1 | Google          | NULL       |
|            1 | Initech         | 2013-01-31 |
|            2 | Woodworking.com | 2016-08-25 |
|            2 | NY Times        | 2013-11-10 |
|            3 | NFL.com         | 2014-04-13 |
+--------------+-----------------+------------+
```

あなたのは、でまだいているが`NULL`をむ、 `2016-01-01`のすべてのをすクエリをすることです。このselectはのとおりです。

```
SELECT * FROM example WHERE end_date > '2016-01-01';
```

`NULL`をつはまれません。

```
+--------------+-----------------+------------+
| applicant_id | company_name    | end_date   |
+--------------+-----------------+------------+
|            2 | Woodworking.com | 2016-08-25 |
+--------------+-----------------+------------+
```

MySQLのドキュメントでは、<、>、=、および<>をしてすると、ブール`TRUE`または`FALSE`わりに `NULL` `FALSE`。 したがって、 `NULL` end_dateをつは、2016-01-01よりきくなく、2016-01-01よりもさいものではありません。

これは、キーワードIS NULLをしてできます。

```
SELECT * FROM example WHERE end_date > '2016-01-01' OR end_date IS NULL;

+--------------+-----------------+------------+
| applicant_id | company_name    | end_date   |
+--------------+-----------------+------------+
|            1 | Google          | NULL       |
```

```
|            2 | Woodworking.com | 2016-08-25 |
+--------------+-----------------+------------+
```

MAX()やGROUP BYなどのがタスクにまれている、NULLのはよりになります。あなたのタスクがapplicant_idののをした、のクエリはなのみにえます。

```
SELECT applicant_id, MAX(end_date) FROM example GROUP BY applicant_id;

+--------------+---------------+
| applicant_id | MAX(end_date) |
+--------------+---------------+
|            1 | 2013-01-31    |
|            2 | 2016-08-25    |
|            3 | 2014-04-13    |
+--------------+---------------+
```

しかし、がにまだされていることをNULLすことがわかっている、ののはです。 CASE WHENをすると、 NULLのがされNULL。

```
 SELECT
    applicant_id,
    CASE WHEN MAX(end_date is null) = 1 THEN 'present' ELSE MAX(end_date) END
    max_date
FROM example
GROUP BY applicant_id;

+--------------+------------+
| applicant_id | max_date   |
+--------------+------------+
|            1 | present    |
|            2 | 2016-08-25 |
|            3 | 2014-04-13 |
+--------------+------------+
```

このは、のexampleにって、がにしたをするためにすことができます。

```
SELECT
  data.applicant_id,
  data.company_name,
  data.max_date
FROM (
  SELECT
    *,
    CASE WHEN end_date is null THEN 'present' ELSE end_date END max_date
  FROM example
) data
INNER JOIN (
 SELECT
   applicant_id,
   CASE WHEN MAX(end_date is null) = 1 THEN 'present' ELSE MAX(end_date) END max_date
 FROM
   example
 GROUP BY applicant_id
) j
ON data.applicant_id = j.applicant_id AND data.max_date = j.max_date;
```

```
+--------------+-----------------+------------+
| applicant_id | company_name    | max_date   |
+--------------+-----------------+------------+
|            1 | Google          | present    |
|            2 | Woodworking.com | 2016-08-25 |
|            3 | NFL.com         | 2014-04-13 |
+--------------+-----------------+------------+
```

これらは、MySQLで`NULL`をうのほんのです。

オンラインでスパースまたはデータのをむ https://riptutorial.com/ja/mysql/topic/5866/スパースまたはデータの

39: セレクト

き

SELECTは、1つまたはのテーブルからされたをりすためにされます。

- SELECT DISTINCT [expressions] FROM TableName [WHERE]; ///シンプルセレクト
- SELECT DISTINCTa、b ...はSELECT DISTINCT a、bとじです。
- SELECT [すべて| DISTINCT | DISTINCTROW] [HIGH_PRIORITY] [STRAIGHT_JOIN] [SQL_SMALL_RESULT | SQL_BIG_RESULT] [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]のFROM[WHERE] [GROUP BY] [HAVING] [ORDER BY[ASC | DESC]] [LIMIT [オフセット] number_rows | LIMIT number_rows OFFSET offset_value] [プロシージャプロシージャ] [INTO [OUTFILE 'file_name' options | DUMPFILE 'file_name' | @ variable1、@ variable2、... @variable_n] [FOR UPDATE |モードでロック]; ///

MySQLのSELECTのについては、 MySQL Docsをしてください。

Examples

による

```
CREATE TABLE stack(
    id INT,
    username VARCHAR(30) NOT NULL,
    password VARCHAR(30) NOT NULL
);

INSERT INTO stack (`id`, `username`, `password`) VALUES (1, 'Foo', 'hiddenGem');
INSERT INTO stack (`id`, `username`, `password`) VALUES (2, 'Baa', 'verySecret');
```

クエリ

```
SELECT id FROM stack;
```

```
+------+
| id   |
+------+
|    1 |
|    2 |
+------+
```

すべてのを*

クエリ

```
SELECT * FROM stack;
```

```
+------+----------+----------+
| id   | username | password |
+------+----------+----------+
|    1 | admin    | admin    |
|    2 | stack    | stack    |
+------+----------+----------+
2 rows in set (0.00 sec)
```

のようにして、の1つのからすべてのをできます。

```
SELECT stack.* FROM stack JOIN Overflow ON stack.id = Overflow.id;
```

ベストプラクティスはしないでください*あなたがにをデバッグしたりフェッチされていないり、そうでないは、スキ―マのは、をべえる/ / DROPをADDなアプリケ―ションエラ―がするがあり、。また、セットになカラムのリストをえると、MySQLのクエリプランナ―はしばしばクエリをできます。

1. を/するときに、 `SELECT *` したをするはありません。
2. それはくのがくて
3. えもされるので、 `SELECT *` -usageはされますか

1. あなたはのデ―タをしています。ごとに200kをむVARBINARYをするとします。このデ―タは、1レコ―ドにつき1つのにしかではありません`SELECT *`をすると、でない10につき2MBをすことになります
2. されるデ―タについての
3. をすると、がされたときにエラ―がする
4. クエリプロセッサはもうしをしなければなりません。テ―ブルにどのようながするのかをするがあります@vinodadhikaryにします
5. がよりにされるをつけることができます
6. SELECT *をすると、すべてのがされます。
7. はにすることはできませんただし、をするのはいです
8. `TEXT`フィ―ルドをするなクエリでは、でないテ―ブルによってクエリがくなることがあります

WHEREをしたSELECT

クエリ

```
SELECT * FROM stack WHERE username = "admin" AND password = "admin";
```

```
+------+----------+----------+
| id   | username | password |
```

```
+------+----------+----------+
|    1 | admin    | admin    |
+------+----------+----------+
1 row in set (0.00 sec)
```

WHEREでネストされたSELECTをしたクエリ

WHEREには、よりなクエリをくためののなSELECTをめることができます。これは 'ネストされた'クエリです

クエリ

ネストされたクエリは、、クエリののアトミックをのためにすためにされます。

```
SELECT title FROM books WHERE author_id = (SELECT id FROM authors WHERE last_name = 'Bar' AND
first_name = 'Foo');
```

メールアドレスのないすべてのユーザーをします。

```
SELECT * FROM stack WHERE username IN (SELECT username FROM signups WHERE email IS NULL);
```

セットをするときに 、パフォーマンスのにジョインをすることをしてください。

LIKEをうSELECT

```
CREATE TABLE stack
(  id int AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(100) NOT NULL
);

INSERT stack(username) VALUES
('admin'),('k admin'),('adm'),('a adm b'),('b XadmY c'), ('adm now'), ('not here');
```

どこでも "adm"

```
SELECT * FROM stack WHERE username LIKE "%adm%";
+----+-----------+
| id | username  |
+----+-----------+
|  1 | admin     |
|  2 | k admin   |
|  3 | adm       |
|  4 | a adm b   |
|  5 | b XadmY c |
|  6 | adm now   |
+----+-----------+
```

"adm"でまります

```
SELECT * FROM stack WHERE username LIKE "adm%";
+----+----------+
| id | username |
+----+----------+
|  1 | admin    |
|  3 | adm      |
|  6 | adm now  |
+----+----------+
```

"adm"でわります

```
SELECT * FROM stack WHERE username LIKE "%adm";
+----+----------+
| id | username |
+----+----------+
|  3 | adm      |
+----+----------+
```

`LIKE`の`%`がののとするのとに、 _はただ1つのとします。えば、

```
SELECT * FROM stack WHERE username LIKE "adm_n";
+----+----------+
| id | username |
+----+----------+
|  1 | admin    |
+----+----------+
```

パフォーマンスノート `username`にインデックスがある、

- `LIKE 'adm'`は `= 'adm'とじことをします
- `LIKE 'adm%'`は`BETWEEN..AND..`の「」 `BETWEEN..AND..`のインデックスをうまくすることができます。
- `LIKE '%adm'` またはワイルドカードをするのは、のインデックスをすることができません。したがって、それはくなります。がいでは、がいがあり、です。
- `RLIKE REGEXP` は`LIKE`よりもくなるがありますが、よりくのがあります。
- MySQLがしているが`FULLTEXT`テーブルとのくののインデックスを、それらの`FULLTEXT`インデックスをしてクエリするためにされていない `LIKE`。

きセレクト AS

SQLエイリアスは、テーブルまたはのをにするためにされます。これらはにをさせるためにされます。

クエリ

```
SELECT username AS val FROM stack;
SELECT username val FROM stack;
```

`AS`はにオプションです。

```
+-------+
| val   |
+-------+
| admin |
| stack |
+-------+
2 rows in set (0.00 sec)
```

LIMITをむSELECT

クエリ

```
SELECT *
  FROM Customers
 ORDER BY CustomerID
 LIMIT 3;
```

ID				シティ		
1	アルフレッド·フッタキステ	マリアアンダース	Obere Str。 57	ベルリン	12209	ドイツ
2	アナトラジ—ロEmparedados y helados	アナトラヒヨ	Avda。デラコンスティチオン2222	メキシコDF	05021	メキシコ
3	アントニオ·モレノ·テケリア	アントニオ·モレノ	マタデロス2312	メキシコDF	05023	メキシコ

ベストプラクティス `LIMIT`するはに`ORDER BY`をしてください。そうしないと、するはできなくなります。

クエリ

```
SELECT *
  FROM Customers
 ORDER BY CustomerID
 LIMIT 2,1;
```

`LIMIT`に2つのがまれている、 `LIMIT` `LIMIT offset,count`としてされ`LIMIT offset,count` 。 したがって、このでは、2つのレコ—ドをスキップして1つをします。

ID				シティ		
3	アントニオ·モレノ·テケリア	アントニオ·モレノ	マタデロス2312	メキシコDF	05023	メキシコ

`LIMIT`のはでなければなりません。のではないがあります。

DISTINCTをしたSELECT

SELECTのDISTINCTは、セットからをDISTINCT。

```
CREATE TABLE `car`
(   `car_id` INT UNSIGNED NOT NULL PRIMARY KEY,
    `name` VARCHAR(20),
    `price` DECIMAL(8,2)
);

INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (1, 'Audi A1', '20000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (2, 'Audi A1', '15000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (3, 'Audi A2', '40000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (4, 'Audi A2', '40000');

SELECT DISTINCT `name`, `price` FROM CAR;
+---------+----------+
| name    | price    |
+---------+----------+
| Audi A1 | 20000.00 |
| Audi A1 | 15000.00 |
| Audi A2 | 40000.00 |
+---------+----------+
```

DISTINCT、々のではなく、をするために、すべてのにわたってします。は、しばしばしいSQLのです。するに、セットのレベルでのは、レベルでのではなくである。これをするには、のセットの "Audi A1"をてください。

MySQLのそれのバージョンでは、 DISTINCTはORDER BYとにすることにをちます。ONLY_FULL_GROUP_BYのは、のMySQLマニュアルページの「 GROUP BYのMySQL 」にされているようにします。

LIKE_をむSELECT

LIKEパターンの_は、1にします。

クエリ

```
SELECT username FROM users WHERE users LIKE 'admin_';
```

```
+----------+
| username |
+----------+
| admin1   |
| admin2   |
| admin-   |
| adminA   |
+----------+
```

CASEまたはIFでのSELECT

クエリ

```
SELECT st.name,
       st.percentage,
       CASE WHEN st.percentage >= 35 THEN 'Pass' ELSE 'Fail' END AS `Remark`
FROM student AS st ;
```

```
+--------------------------------+
|   name   | percentage | Remark |
+--------------------------------+
|   Isha   |     67     |  Pass  |
|   Rucha  |     28     |  Fail  |
|   Het    |     35     |  Pass  |
|   Ansh   |     92     |  Pass  |
+--------------------------------+
```

または**IF**

```
SELECT st.name,
       st.percentage,
       IF(st.percentage >= 35, 'Pass', 'Fail') AS `Remark`
FROM student AS st ;
```

NB

```
IF(st.percentage >= 35, 'Pass', 'Fail')
```

> これはのことをしますIF st.percentage> = 35が**TRUE**ならば、`'Pass'`しますELSEは`'Fail'`

の**SELECT**

BETWEENをして、「よりきいANDよりさいしい」のみわせをきえることができます。

データ

```
+----+-----------+
| id | username  |
+----+-----------+
|  1 | admin     |
|  2 | root      |
|  3 | toor      |
|  4 | mysql     |
|  5 | thanks    |
|  6 | java      |
+----+-----------+
```

オペレータによるクエリ

```
SELECT * FROM stack WHERE id >= 2 and id <= 5;
```

BETWEENとのクエリ

```
SELECT * FROM stack WHERE id BETWEEN 2 and 5;
```

```
+----+-----------+
| id | username  |
+----+-----------+
|  2 | root      |
|  3 | toor      |
|  4 | mysql     |
|  5 | thanks    |
+----+-----------+
4 rows in set (0.00 sec)
```

BETWEENは`>=`と`<=`、not `>`と`<`ます。

NOT BETWEENをする

ネガをするは`NOT`をできます。えば

```
SELECT * FROM stack WHERE id NOT BETWEEN 2 and 5;
```

```
+----+-----------+
| id | username  |
+----+-----------+
|  1 | admin     |
|  6 | java      |
+----+-----------+
2 rows in set (0.00 sec)
```

のにない`>`と`<`なく`>=`と`<=`すなわち、 `WHERE id NOT BETWEEN 2 and 5`とじである`WHERE (id < 2 OR id > 5)`。

`BETWEEN`でするカラムにインデックスがある、MySQLはそのインデックスをレンジスキャンにできます。

をむ**SELECT**

```
SELECT ... WHERE dt >= '2017-02-01'
             AND dt  < '2017-02-01' + INTERVAL 1 MONTH
```

かに、これは`BETWEEN`と`23:59:59`みみでうことができます。しかし、このパターンにはのがあります。

- をにするはありませんからなさになることがい
- のエンドポイント `BETWEEN`ようにをめず、また '235959'としないでください。
- これは、 `DATE` 、 `TIMESTAMP` 、 `DATETIME` 、さらにはマイクロにまれる`DATETIME(6)`ます。
- 、などのをいます。
- それはインデックスにしい `BETWEEN`。

オンラインでセレクトをむ https://riptutorial.com/ja/mysql/topic/3307/セレクト

40: タイムゾーンの

MySQLのなユーザーベースのをするがあるは、テーブルにTIMESTAMPデータをします。

ユーザーごとに、ユーザータイムゾーンをします。 VARCHAR64は、そののなデータです。ユーザーがシステムにすると、タイムゾーンのをねます。のものはAtlantic Time、 `America/Edmonton`です。あなたは、 `Asia/Kolkata`または`Australia/NSW`もあれば、そうでないもあります。このユーザーのユーザーインターフェイスには、WordPress.orgソフトウェアがいです。

に、ユーザーのわりにホストプログラムJava、PHPなどからDBMSへのをするたびに、SQLコマンドをします

```
SET SESSION time_zone='(whatever tz string the user gave you)'
```

ユーザーデータのにがかかることがあります。インストールした`TIMESTAMP`はすべて、ユーザーのでされます。

これにより、テーブルにるすべてのがUTCにされ、すべてのがローカルにされます。それはNOWとCURDATEにしてしくします。このも、DATETIMEまたはDATEデータではなく、TIMESTAMPをするがあります。

サーバーのOSとデフォルトのMySQLタイムゾーンがUTCにされていることをしてください。をデータベースにロードするにこれをわなければ、することはほとんどです。ベンダーをしてMySQLをするは、このをすることをします。

Examples

のタイムゾーンでのとをします。

これにより、、インド、およびUTCでの`NOW()`がされます。

```
SELECT NOW();
SET time_zone='Asia/Kolkata';
SELECT NOW();
SET time_zone='UTC';
SELECT NOW();
```

された`DATE`または`DATETIME`をのタイムゾーンにします。

されている`DATE`または`DATETIME` カラムのどこかにあるがあるタイムゾーンにしてされていたのですが、MySQLではタイムゾーンにがされていません 。したがって、のタイムゾーンにするはのタイムゾーンをっているがあります。 `CONVERT_TZ()`をするとがわれます。このは、でカリフォルニアでされているをしています。

```
SELECT CONVERT_TZ(date_sold,'UTC','America/Los_Angeles') date_sold_local
  FROM sales
 WHERE state_sold = 'CA'
```

のタイムゾーンでされた`TIMESTAMP`をする

これはにです。すべてのTIMESTAMPはでされ、レンダリングされるたびににのtime_zoneにされます。

```
SET SESSION time_zone='America/Los_Angeles';
SELECT timestamp_sold
  FROM sales
 WHERE state_sold = 'CA'
```

どうしてこれなの TIMESTAMPは、しいUNIXのtime_tデータにづいています 。これらのUNIXタイムスタンプは、 1970-01-01 00:00:00 UTCのでされ1970-01-01 00:00:00。

TIMESTAMPはでされTIMESTAMP。 DATEとDATETIMEは、されているになくされます。

サーバーのローカルタイムゾーンのはですか

サーバーには、サーバーマシンのによってされたデフォルトのグローバルtime_zoneがあります。このでのタイムゾーンをすることができます

```
SELECT @@time_zone
```

なことに、これは、SYSTEMします。つまり、MySQLのはサーバーOSのタイムゾーンによってされます。

こののクエリはい、 ハック は、サーバーのタイムゾーンとUTCののオフセットをでします。

```
CREATE TEMPORARY TABLE times (dt DATETIME, ts TIMESTAMP);
SET time_zone = 'UTC';
INSERT INTO times VALUES(NOW(), NOW());
SET time_zone = 'SYSTEM';
SELECT dt, ts, TIMESTAMPDIFF(MINUTE, dt, ts)offset FROM times;
DROP TEMPORARY TABLE times;
```

これはどのようにしますかなるデータをつテーブルの2つのがヒントです。 DATETIMEデータはにでテーブルにされ、 TIMESTAMPはUTCでされTIMESTAMP。したがって、time_zoneがUTCにされたときにされるINSERTは、2つのの/をします。

に、SELECTステートメントは、time_zoneがサーバーのにされているときにされます。TIMESTAMPはに、されているUTCからSELECTステートメントのローカルにされます。 DATETIMEはそうではありません。したがって、 TIMESTAMPDIFF(MINUTE...)は、ローカルとのをします。

サーバーでできるtime_zoneのはですか

MySQLサーバインスタンスでtime_zoneのリストをするには、このコマンドをします。

```
SELECT mysql.time_zone_name.name
```

、これは、Paul Eggertによって Internet Assigned Numbers Authorityでされているタイムゾーンの ZoneInfoリストをしています 。に600のタイムゾーンがあります。

UNIXのようなオペレーティングシステムLinuxディストリビューション、BSDディストリビューション、のMac OSディストリビューションなどは、なアップデートをけります。これらのをオペレーティングシステムにインストールすると、そこでされているMySQLインスタンスが、タイムゾーンおよびのをできます。

はるかにいタイムゾーンのリストをすると、サーバーがにされているか、Windowsでされています。 ここでされている zoneinfoのリストをインストールし、するために、サーバーのために。

オンラインでタイムゾーンのをむ https://riptutorial.com/ja/mysql/topic/7849/タイムゾーンの

41: データベースの

- CREATE {DATABASE | SCHEMA} [しない] db_name [create_specification] ///データベースをするには
- DROP {DATABASE | SCHEMA} [IF EXISTS] db_name ///データベースをするには

パラメーター

パラメータ	
CREATE DATABASE	されたのデータベースをします。
スキーマの	これは`CREATE DATABASE`です
しない	されたデータベースがすでにする、エラーをするためにされます
create_specification	`create_specification`オプションは、`CHARACTER SET`や`COLLATE`データベースなどのデータベースをします。

Examples

データベース、ユーザー、およびの

DATABASEをします。されたSCHEMAはとしてできることにしてください。

```
CREATE DATABASE Baseball; -- creates a database named Baseball
```

データベースがすでにするは、エラー1007がされます。このエラーをするには、をしてください。

```
CREATE DATABASE IF NOT EXISTS Baseball;
```

に、

```
DROP DATABASE IF EXISTS Baseball; -- Drops a database if it exists, avoids Error 1008
DROP DATABASE xyz; -- If xyz does not exist, ERROR 1008 will occur
```

のエラーののため、DDLステートメントは`IF EXISTS`よくされます。

デフォルトのCHARACTER SETとをしてデータベースをできます。えば

```
CREATE DATABASE Baseball CHARACTER SET utf8 COLLATE utf8_general_ci;
```

```
SHOW CREATE DATABASE Baseball;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| Baseball | CREATE DATABASE `Baseball` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
```

あなたののデ―タベ―スをしてください

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ajax_stuff         |
| Baseball           |
+--------------------+
```

アクティブなデ―タベ―スをし、いくつかのをしてください

```
USE Baseball; -- set it as the current database
SELECT @@character_set_database as cset,@@collation_database as col;
+------+-----------------+
| cset | col             |
+------+-----------------+
| utf8 | utf8_general_ci |
+------+-----------------+
```

は、デ―タベ―スのデフォルトのセットとをしています。

ユ―ザ―をする

```
CREATE USER 'John123'@'%' IDENTIFIED BY 'OpenSesame';
```

のでは、ユ―ザ―John123がされ、 %ワイルドカ―ドのためにのホストにできます。ユ―ザ―のパスワ―ドは、ハッシュされた 'OpenSesame'にされています。

そしてのものをりなさい

```
CREATE USER 'John456'@'%' IDENTIFIED BY 'somePassword';
```

な mysqlデ―タベ―スをべてユ―ザがされたことをします

```
SELECT user,host,password from mysql.user where user in ('John123','John456');
+---------+------+-------------------------------------------+
| user    | host | password                                  |
+---------+------+-------------------------------------------+
| John123 | %    | *E6531C342ED87 .....................      |
| John456 | %    | *B04E11FAAAE9A .....................      |
+---------+------+-------------------------------------------+
```

こので、ユーザーはされていますが、Baseballデータベースをするはありません。

ユーザーとデータベースのアクセスをしてします。ユーザーJohn123にBaseballデータベースにするなをえ、のユーザーのSELECTのみをする

```
GRANT ALL ON Baseball.* TO 'John123'@'%';
GRANT SELECT ON Baseball.* TO 'John456'@'%';
```

をしてください

```
SHOW GRANTS FOR 'John123'@'%';
+-----------------------------------------------------------------------------------------------
-----------+
| Grants for John123@%
|
+-----------------------------------------------------------------------------------------------
-----------+
| GRANT USAGE ON *.* TO 'John123'@'%' IDENTIFIED BY PASSWORD '*E6531C342ED87
.....................        |
| GRANT ALL PRIVILEGES ON `baseball`.* TO 'John123'@'%'
|
+-----------------------------------------------------------------------------------------------
-----------+

SHOW GRANTS FOR 'John456'@'%';
+-----------------------------------------------------------------------------------------------
-----------+
| Grants for John456@%
|
+-----------------------------------------------------------------------------------------------
-----------+
| GRANT USAGE ON *.* TO 'John456'@'%' IDENTIFIED BY PASSWORD '*B04E11FAAAE9A
.....................        |
| GRANT SELECT ON `baseball`.* TO 'John456'@'%'
|
+-----------------------------------------------------------------------------------------------
-----------+
```

あなたがにることができる`GRANT USAGE`のは、にユーザーがログインできることをすることにしてください。それだけでがあります。

MyDatabase

あなたは、のデータベースをし、のデータベースのいずれかにきみをしてはいけません。これは、めてしたにうのの1つになるがいです。

```
CREATE DATABASE my_db;
USE my_db;
CREATE TABLE some_table;
INSERT INTO some_table ...;
```

データベース`my_db.some_table`することで、テーブルをできます。

システムデータベース

MySQLののためにのデータベースがします。それらをみる `SELECT` ことはできますが、そのにテーブルをきむ `INSERT` / `UPDATE` / `DELETE` しないでください。 いくつかのがあります。

- `mysql` - `GRANT`やそののもののためのリポジトリ。
- `information_schema` - ここでのは、にはメモリによってされているというでは「」です。そのには、すべてののスキーマがまれます。
- `performance_schema` - ?? [してから、してください]
- の MariaDB、Galera、TokuDBなど

データベースのと

をするにがデータベースをした、そのデータベースをすることができます。それのは、でするがあります。

```
mysql> CREATE DATABASE menagerie;
```

Unixでは、データベースはとがされますSQLキーワードとはなりますので、データベースをMenagerie、MENAGERIE、またはそののではなく、にmenagerieとしてするがあります。これはテーブルにもてはまります。 Windowsでは、このはされませんが、のクエリでじをしてデータベースとテーブルをするがありますが、さまざまなにより、にされるベストプラクティスは、データベースがされました。

データベースをしても、それをするためにはされません。それをにうがあります。 menagerieをのデータベースにするには、のステートメントをします。

```
mysql> USE menagerie
Database changed
```

データベースはするがありますが、mysqlセッションをするたびにするようにするがあります。こののようにUSEをすると、これをうことができます。あるいは、mysqlをびすときにコマンドラインでデータベースをすることもできます。するのあるパラメータのにそのをするだけです。えば

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

オンラインでデータベースのをむ https://riptutorial.com/ja/mysql/topic/600/データベースの

42: データ

Examples

/

```
select '123' * 2;
```

2 するために、MySQLはに123をにします。

り

246

へのは、からにまります。がな、は0

```
select '123ABC' * 2
```

り

246

```
select 'ABC123' * 2
```

り

0

VARCHAR255 - そうでないか

される**len**

に、に16であるか、あるいはASCIIにされているいくつかのなについてします。これらのために、スペースをにしないようにCHARACTER SET ascii latin1はですをするがあります

```
UUID CHAR(36) CHARACTER SET ascii -- or pack into BINARY(16)
country_code CHAR(2) CHARACTER SET ascii
ip_address CHAR(39) CHARACTER SET ascii -- or pack into BINARY(16)
phone VARCHAR(20) CHARACTER SET ascii -- probably enough to handle extension
postal_code VARCHAR(20) CHARACTER SET ascii -- (not 'zip_code') (don't know the max

city VARCHAR(100) -- This Russian town needs 91:
    Poselok Uchebnogo Khozyaystva Srednego Professionalno-Tekhnicheskoye Uchilishche Nomer
Odin
country VARCHAR(50) -- probably enough
name VARCHAR(64) -- probably adequate; more than some government agencies allow
```

に255ではないのはなぜですかすべてに255をするなプラクティスをける2つのがあります。

- なSELECTがサブクエリ、 UNION 、 GROUP BYなどのテーブルをするがあるは、 MEMORYエンジンをしてデータをRAMにすることをおめします。しかし、 VARCHARsはプロセスでCHARにされます。これにより、 VARCHAR(255) CHARACTER SET utf8mb4は1020バイトになります。これはディスクにするがありますが、くなります。
- あるでは、InnoDBはテーブルのカラムのなサイズをべ、それがきすぎるとしてCREATE TABLEちります。

VARCHARとTEXT

*TEXT 、 CHAR 、およびVARCHARヒントにえて、いくつかのベスト・プラクティス

- TINYTEXTしてしないでください。
- CHARほとんどしない - です。はCHARACTER SETえば、utf8mb4のは4バイト/です。
- CHARでは、にかっていないり、 CHARACTER SET asciiしてください。
- VARCHAR(n)はn でりてられます。 TEXTは、いくつかのバイトでりてられます 。 しかし、りてがですか
- *TEXT は、 SELECTsテーブルのいによってなSELECTsくすることがあります。

AUTO_INCREMENTとしてのINT

AUTO_INCREMENTは、 INTのサイズをできます。 UNSIGNEDはにです。

のによってAUTO_INCREMENT IDが「きけられる」 AUTO_INCREMENTしてください。これはのギャップにつながるがあります。 INSERT IGNOREおよびREPLACE。それらはではないことをするに IDをにりてるかもしれません。これは、InnoDBエンジンでされるであり、にされており、をげるものではありません。

その

に "FLOAT、DOUBLE、DECIMAL"と "ENUM"のエントリがにあります。データにするのページはいにくいものです - は「フィールド」またはそれを「データ」とぶべきですかをし、にこれらのトピックページにすることをおめします

- INTs
- FLOAT、DOUBLE、およびDECIMAL
- CHAR、TEXTなど
- バイナリとBLOB
- DATETIME、TIMESTAMP、およびフレンド
- ENUMとSET
- データ
- JSONタイプ MySQL 5.7.8
- のデータにshoehorningをとするMoneyやそののなをする

な、トピックペ―ジには、やにえて、のものをめるがあります。

- の
- サイズバイト
- MySQLエンジンとはい
- PRIMARY KEYまたは2キ―でデ―タをするの
- そののベストプラクティス
- そののパフォ―マンスの

のがまたはされたときに、この「」がするとします。

はじめに

MySQLはさまざまなをしています。これらは、

グル―プ	タイプ
	INTEGER、 INT、 SMALLINT、 TINYINT、 MEDIUMINT、 BIGINT
	DECIMAL、 NUMERIC
	FLOAT、 DOUBLE
ビットタイプ	BIT

なしのはに0です。

タイプ	ストレ―ジバイト	された	された	なし
TINYINT	1	-2 7 -128	2 7 -1 127	2 8 -1 255
SMALLINT	2	-2 15 -32,768	2 15 -1 32,767	2 16 -1 65,535
MEDIUMINT	3	-2 23 -8,388,608	2 23 -1 8,388,607	2 24 -1 16,777,215
INT	4	-2 31	2 31 -1	2 32 -1

タイプ	ストレージバイト	された	された	なし
		-2,147,483,648	2,147,483,647	4,294,967,295
BIGINT	8	-2 63 - 9,223,372,036,854,775,808	2 63 -1 9,223,372,036,854,775,807	2 64 -1 18,446,744,073,709,551,61

MySQLのDECIMALとNUMERICは、なデータをします。これらのをして、などなをすることをおめします。

これらのはバイナリでされます。では、とりをするがあります

Precisionは、としてされるをします。

Scaleは、にされたをします。

```
salary DECIMAL(5,2)
```

5はprecisionをし、2はscaleします。このでは、このにできるのは-999.99 to 999.99

scaleパラメータをすると、デフォルトは0になります。

このデータは、65までできます。

DECIMAL(M,N)がとるバイトはおよそ M/2です。

FLOATとDOUBLEはおおよそのデータをします。

タイプ	ストレージ		
く	4バイト	23のビット/7の10	10 ^ + / - 38
ダブル	8バイト	53のビット/ 16の10	10 ^ + / - 308

REALはFLOATです。 DOUBLE PRECISIONですDOUBLE。

MySQLはM、Dもしますが、それをしないでください。 M、Dは、をMまでできることをします.Dはにすることができます。 は2められるかりてられます。これはよりもくのをきこすでしょう。

はでありなとしてされないため、でにうとがするがあります。に、 FLOATはDOUBLEとほとんどじではないことにしてください。

ビットタイプ

BITは、ビットフィールドをするのにです。 BIT(M)は、Mが1 to 64にあるMビットまでのをにするをbit valueでることもできます。

```
b'111'      -> 7
b'10000000' -> 128
```

には、128ビットにして(1 << 7)ように、ビットのをするために 'shift'をするとなことがあります。

NDBテーブルのすべてのBITカラムのサイズは4096です。

CHARn

CHAR(n)はn の のです。それがCHARACTER SET utf8mb4、そのになく、に4*n バイトをめることをします。

CHAR(n)ほとんどのは、をむをんでいるため、 CHARACTER SET asciiなければなりません。 latin1はlatin1です。

```
country_code CHAR(2) CHARACTER SET ascii,
postal_code  CHAR(6) CHARACTER SET ascii,
uuid    CHAR(39) CHARACTER SET ascii,  -- more discussion elsewhere
```

DATE、DATETIME、TIMESTAMP、YEAR、およびTIME

DATEデータにはがまれますが、コンポーネントはまれません。は'YYYY-MM-DD'で、は '1000-01-01' '9999-12-31'です。

DATETIMEには、 'YYYY-MM-DD HHMMSS'というのがまれます。 '1000-01-01 00:00:00'から '9999-12-31 23:59:59'のです。

TIMESTAMPは、 '1970-01-01 00:00:01' UTCから '2038-01-19 03:14:07' UTCまでのをつとをむです。

YEARは、をし、19012155のをします。

TIMEは 'HHMMSS'というのをし、 '-8385959'から '8385959'までのをします。

ストレージ

```
|-----------|--------------------|----------------------------------------|
| Data Type | Before MySQL 5.6.4 | as of MySQL 5.6.4                      |
|-----------|--------------------|----------------------------------------|
| YEAR      |      1 byte        |  1 byte                                |
| DATE      |      3 bytes       |  3 bytes                               |
| TIME      |      3 bytes       |  3 bytes + fractional seconds storage  |
| DATETIME  |      8 bytes       |  5 bytes + fractional seconds storage  |
| TIMESTAMP |      4 bytes       |  4 bytes + fractional seconds storage  |
|-----------|--------------------|----------------------------------------|
```

バージョン5.6.4

```
|------------------------------|------------------|
| Fractional Seconds Precision | Storage Required |
|------------------------------|------------------|
|              0               |      0 bytes     |
|              1,2             |      1 byte      |
|              3,4             |      2 byte      |
|              5,6             |      3 byte      |
|------------------------------|------------------|
```

MySQLのマニュアルページのDATE、DATETIME、およびTIMESTAMPの 、 データのストレージ 、およびののをしてください。

オンラインでデータをむ https://riptutorial.com/ja/mysql/topic/4137/データ

43: テーブル

- CREATE TABLE table_namecolumn_name1のdata_typeサイズ、column_name2のdata_typeサイズ、column_name3のdata_typeサイズ、....; //なテーブルの
- CREATE TABLE table_name [しない]column_name1 data_typesize、column_name2 data_typesize、column_name3 data_typesize、....; //のテーブルチェック
- CREATE [TEMPORARY] TABLE table_name [しない]column_name1 data_typesize、column_name2 data_typesize、column_name3 data_typesize、....; //テンポラリテーブルの
- テーブルをするnew_tbl [AS] SELECT * FROM orig_tbl; // SELECTからのテーブルの

`CREATE TABLE`ステートメントは、 `ENGINE`でわるがあります。

```
CREATE TABLE table_name ( column_definitions ) ENGINE=engine;
```

いくつかのオプションがあります

- `InnoDB` バージョン5.5.5のデフォルトトランスフォームセーフACIDエンジンです。トランザクションのコミットとロールバック、およびクラッシュリカバリとレベルのロックをえています。
- `MyISAM` バージョン5.5.5よりのデフォルトこれはのエンジンです。トランザクションやキーはサポートしていませんが、データウェアハウジングにはです。
- `Memory` になのためにすべてのデータをRAMにしますが、データベースのにテーブルのがわれます。

よりくのエンジンオプションがここにあります 。

Examples

なテーブルの

`CREATE TABLE`ステートメントは、MySQLデータベースにテーブルをするためにされます。

```
CREATE TABLE Person (
    `PersonID`      INTEGER NOT NULL PRIMARY KEY,
    `LastName`      VARCHAR(80),
    `FirstName`     VARCHAR(80),
    `Address`       TEXT,
    `City`          VARCHAR(100)
) Engine=InnoDB;
```

すべてのフィールドにはのものがです。

1. フィールドなフィールド。 `-chars 'にをれてください。これにより、fieldnameにえば

space-charsをできるようになります。
2. データ[さ]フィールドが`CHAR`または`VARCHAR`は、フィールドをすることがです。
3. `NULL` | `NOT NULL` `NOT NULL`をすると、そのフィールドに`NULL`をしようとするとします。
4. データとそののをしてくださいここに 。

`Engine=...`は、テーブルのストレージエンジンをするためのオプションのパラメータです。ストレージエンジンがされていない、サーバのデフォルトテーブルストレージエンジンはInnoDBまたはMyISAMをしてテーブルがされます。

デフォルト

さらに、 `DEFAULT`をしてフィールドのデフォルトをすることができます。

```
CREATE TABLE Address (
    `AddressID`   INTEGER NOT NULL PRIMARY KEY,
    `Street`      VARCHAR(80),
    `City`        VARCHAR(80),
    `Country`     VARCHAR(80) DEFAULT "United States",
    `Active`      BOOLEAN DEFAULT 1,
) Engine=InnoDB;
```

に`Street`がされていない、そのフィールドはりされると`NULL`なります。に`Country`がされていない、デフォルトは "United States"になります。

`BLOB` 、 `TEXT` 、 `GEOMETRY` 、および`JSON`フィールドをくすべてのタイプのデフォルトをできます。

キーによるテーブルの

```
CREATE TABLE Person (
    PersonID     INT UNSIGNED NOT NULL,
    LastName     VARCHAR(66) NOT NULL,
    FirstName    VARCHAR(66),
    Address      VARCHAR(255),
    City         VARCHAR(66),
    PRIMARY KEY (PersonID)
);
```

キーは、のをにする`NOT NULL`またはのIDです。 インデックスがされ、 `NOT NULL`としてにされていない、MySQLはににします。

には`PRIMARY KEY`は1つしかなく、には1つの`PRIMARY KEY`があることがされています。 InnoDBはこれがのときににします MySQLのマニュアルにられるように。

しばしば、 "サロゲートキー"ともばれる`AUTO_INCREMENT INT`は、いインデックスのやのテーブルとのけにされます。このは、しいレコードがされるたびには1ずつし、デフォルトの1からまります。

しかし、そのにもかかわらず、があることをするのはではなく、なるかつユニークなものであ

る。

TRUNCATE TABLEをしてをりてないり、のすべてのがされると、INTはデフォルトのにリセットされません。

1つのをキー—インライインとしてする

キー—がのでされている、 PRIMARY KEYはとともにインライインにできます。

```
CREATE TABLE Person (
    PersonID      INT UNSIGNED NOT NULL PRIMARY KEY,
    LastName      VARCHAR(66) NOT NULL,
    FirstName     VARCHAR(66),
    Address       VARCHAR(255),
    City          VARCHAR(66)
);
```

こののコマンドはくてみやすい。

のキー—の

のをむキー—をすることもできます。これは、えば、キー—のテー—ブルでうことができます。するを々のPRIMARY KEYにリストすることによって、のキー—がされます。インラインはここではされていません。これは、 PRIMARY KEYインライインでされるが1つだけであるためです。えば

```
CREATE TABLE invoice_line_items (
    LineNum       SMALLINT UNSIGNED NOT NULL,
    InvoiceNum    INT UNSIGNED NOT NULL,
    -- Other columns go here
    PRIMARY KEY (InvoiceNum, LineNum),
    FOREIGN KEY (InvoiceNum) REFERENCES -- references to an attribute of a table
);
```

キー—のは、ソー—トでするがあります。これは 、ののように、がされたとはなるがあります。

インデックスがきくなると、ディスク、メモリ、およびI/Oがえます。したがって、キー—はなりさくするがありますにキー—にして。 InnoDBでは、すべての「セカンダリインデックス」には、PRIMARY KEYのコピー—がまれています。

キー—によるテー—ブルの

```
CREATE TABLE Account (
    AccountID      INT UNSIGNED NOT NULL,
    AccountNo      INT UNSIGNED NOT NULL,
    PersonID     INT UNSIGNED,
    PRIMARY KEY (AccountID),
    FOREIGN KEY (PersonID) REFERENCES Person (PersonID)
```

```
) ENGINE=InnoDB;
```

キーキー FK は、 テーブルののまたはのです。このFKは、 テーブルにすることがされています。FKするテーブルキーはキーであるが、それはされないことをくおめします。これは、であるがないへのファーストルックアップとしてされ、にそこでものインデックスになることができます。

キーのには、データをするテーブルと、そのをすじをつテーブルがあります。 FOREIGN KEYは、テーブルにされています。テーブルとテーブルは、じストレージエンジンをするがあります。TEMPORARYテーブルであってはなりません。

キーとキーのするには、のデータがです。のサイズとはじでなければなりません。のさはじであるはありません。バイナリの、セットとはじでなければなりません。

キーは、InnoDBストレージエンジンMyISAMまたはMEMORYではなくでサポートされています。のエンジンをするDBセットアップはこのCREATE TABLEステートメントをけれますが、キーをしません。 しいMySQLのバージョンはデフォルトでInnoDBになっていますが、にすることをおめします。

のテーブルのクローン

テーブルはのようにできます。

```
CREATE TABLE ClonedPersons LIKE Persons;
```

しいは、およびをむのとまったくじをちます。

でテーブルをするだけでなく、のテーブルからデータをしてテーブルをすることもできます。

```
CREATE TABLE ClonedPersons SELECT * FROM Persons;
```

SELECTステートメントののいずれかをして、データをすることができます。

```
CREATE TABLE ModifiedPersons
SELECT PersonID, FirstName + LastName AS FullName FROM Persons
WHERE LastName IS NOT NULL;
```

SELECTからテーブルをする、キーとインデックスはされません。あなたはそれらをするがあります

```
CREATE TABLE ModifiedPersons (PRIMARY KEY (PersonID))
SELECT PersonID, FirstName + LastName AS FullName FROM Persons
WHERE LastName IS NOT NULL;
```

SELECTからのテーブルの

CREATE TABLEステートメントのにSELECTステートメントをすることによって、のテーブルをするこ

とができます。

```
CREATE TABLE stack (
    id_user INT,
    username VARCHAR(30),
    password VARCHAR(30)
);
```

じデータベースにテーブルをする

```
-- create a table from another table in the same database with all attributes
CREATE TABLE stack2 AS SELECT * FROM stack;

-- create a table from another table in the same database with some attributes
CREATE TABLE stack3 AS SELECT username, password FROM stack;
```

なるデータベースからテーブルをする

```
-- create a table from another table from another database with all attributes
CREATE TABLE stack2 AS SELECT * FROM second_db.stack;

-- create a table from another table from another database with some attributes
CREATE TABLE stack3 AS SELECT username, password FROM second_db.stack;
```

NB

のデータベースにするのテーブルとじテーブルをするには、のようにデータベースのをするがあります。

```
FROM NAME_DATABASE.name_table
```

テーブルをする

テーブルのスキーマをするには、のいずれかをします。

```
SHOW CREATE TABLE child; -- Option 1

CREATE TABLE `child` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fullName` varchar(100) NOT NULL,
  `myParent` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `mommy_daddy` (`myParent`),
  CONSTRAINT `mommy_daddy` FOREIGN KEY (`myParent`) REFERENCES `parent` (`id`)
      ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

mysqlコマンドラインツールからする、これはあまりではありません。

```
SHOW CREATE TABLE child \G
```

のをすためのではないです。

```
mysql> CREATE TABLE Tab1(id int, name varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> DESCRIBE Tab1; -- Option 2

+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | YES  |     | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

DESCRIBEと**DESC**のでじがられます。

にデータベースのすべてのテーブルで`DESCRIBE`するには、このをしてください。

タイムスタンプをしてテーブルをし、をする

TIMESTAMPには、がにされたがされます。

```
CREATE TABLE `TestLastUpdate` (
    `ID` INT NULL,
    `Name` VARCHAR(50) NULL,
    `Address` VARCHAR(50) NULL,
    `LastUpdate` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
COMMENT='Last Update'
;
```

オンラインでテーブルをむ https://riptutorial.com/ja/mysql/topic/795/テーブル

44: トランザクション

Examples

トランザクションの

トランザクションは、select、insert、update、deleteなどのSQLのシーケンシャルなグループであり、のワークユニットとしてされます。

つまり、グループの々のがしないり、トランザクションはしません。トランザクションでがすると、トランザクションがします。

これをするためののは、です。 2つののをしてください。これをするには、のことをうSQLをするがあります

1. のアカウントでリクエストされたのをする
2. のアカウントからをしてください
3. それを2のアカウントにする

これらのプロセスがしたは、をのにすがあります。

ACIDトランザクションのプロパティ

トランザクションにはの4つのプロパティがあります

- アトミックのすべてのがにしたことをします。それの、トランザクションはにされ、のはのにロールバックされます。
- トランザクションがにコミットされたときに、データベースがをにするようにします。
- トランザクションをいにして、ににさせることができます。
- システムにがしたでも、コミットされたトランザクションのまたはがにされます。

トランザクションは`START TRANSACTION`または`BEGIN WORK`まり、 `COMMIT`または`ROLLBACK`でします。とののSQLコマンドは、トランザクションのをします。

```
START TRANSACTION;
SET @transAmt = '500';
SELECT @availableAmt:=ledgerAmt FROM accTable WHERE customerId=1 FOR UPDATE;
UPDATE accTable SET ledgerAmt=ledgerAmt-@transAmt WHERE customerId=1;
UPDATE accTable SET ledgerAmt=ledgerAmt+@transAmt WHERE customerId=2;
COMMIT;
```

`START TRANSACTION`では、 `COMMIT`または`ROLLBACK`してトランザクションをするまで、コミットはのまま`ROLLBACK`。 コミットモードは、それまでのにります。

`FOR UPDATE`は、トランザクションののをしますおよびロックします。

トランザクションはコミットのままですが、このトランザクションはのユーザーにはできません。

にわるなき

- SQLコマンド`BEGIN WORK`または`START TRANSACTION`して、トランザクションをします。
- すべてのSQLをします。
- あなたのにしたがってすべてがされているかどうかをします。
- はいのは`COMMIT`コマンドをし、それのはすべてをのにす`ROLLBACK`コマンドをし`ROLLBACK`。
- Galera / PXCをしている、またはにしているがあるは、 `COMMIT`もエラーをチェックしてください。

COMMIT、ROLLBACKおよびAUTOCOMMIT

AUTOCOMMIT

MySQLは、トランザクションのではないをにコミットします。 `BEGIN`または`START TRANSACTION`がにいていない`UPDATE` 、 `DELETE`または`INSERT`ステートメントのは、すべてのでちにされます。

`AUTOCOMMIT`は、デフォルトで*true*にされます。これはのようにできます。

```
--->To make autcommit false
SET AUTOCOMMIT=false;
--or
SET AUTOCOMMIT=0;

--->To make autcommit true
SET AUTOCOMMIT=true;
--or
SET AUTOCOMMIT=1;
```

`AUTOCOMMIT`ステータスをするには

```
SELECT @@autocommit;
```

コミット

`AUTOCOMMIT`がfalseにされ、トランザクションがコミットされていない、はのにしてのみされます。

`COMMIT`がテーブルへのをコミットすると、はすべてのでされます。

これをする2つのをえます

1

```
--->Before making autocommit false one row added in a new table
mysql> INSERT INTO testTable VALUES (1);
```

```
--->Making autocommit = false
mysql> SET autocommit=0;

mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+
```

2

```
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
+-----+
---> Row inserted before autocommit=false only visible here
```

1

```
mysql> COMMIT;
--->Now COMMIT is executed in connection 1
mysql> SELECT * FROM testTable;
    +-----+
    | tId |
    +-----+
    |   1 |
    |   2 |
    |   3 |
    +-----+
```

2

```
mysql> SELECT * FROM testTable;
    +-----+
    | tId |
    +-----+
    |   1 |
    |   2 |
    |   3 |
    +-----+
--->Now all the three rows are visible here
```

ロールバック

クエリのにかがじたは、 `ROLLBACK`をしてをにします。のをしてください

```
--->Before making autocommit false one row added in a new table
mysql> INSERT INTO testTable VALUES (1);
```

```
--->Making autocommit = false
mysql> SET autocommit=0;

mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+
```

たちは`ROLLBACK`をしてい`ROLLBACK`

```
--->Rollback executed now
mysql> ROLLBACk;

mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
+-----+
--->Rollback removed all rows which all are not committed
```

`COMMIT`がされると、 `ROLLBACK`はも`ROLLBACK`ない

```
mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
mysql> COMMIT;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+

--->Rollback executed now
mysql> ROLLBACk;

mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+
--->Rollback not removed any rows
```

`AUTOCOMMIT`が*true*にされている 、 `COMMIT`と`ROLLBACK`はにたない

JDBCドライバをしたトランザクション

JDBCドライバをしたトランザクションは、トランザクションのコミットとロールバックのとタイミングをするためにされます。 MySQLサーバへのは、JDBCドライバをしてされます。

MySQLJDBCドライバはここからダウンロードできます

JDBCドライバをしてデータベースにすることからめることができます

```
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection(DB_CONNECTION_URL,DB_USER,USER_PASSWORD);
--->Example for connection url "jdbc:mysql://localhost:3306/testDB");
```

セット これは、クライアントがSQLステートメントをサーバにするためにするセットをします。また、サーバーがをクライアントにりすためにするセットもします。

これは、サーバーへのをするにするがあります。だから、は、

```
jdbc:mysql://localhost:3306/testDB?useUnicode=true&amp;characterEncoding=utf8
```

セットとのは、これをしてください。

をくと、 `AUTOCOMMIT`モードはデフォルトで*true*にされ、トランザクションをするには*false*にするがあります。

```
con.setAutoCommit(false);
```

をいたにはに`setAutoCommit()`メソッドをびすがあります。

それのは、 `START TRANSACTION`または`BEGIN WORK`をしてしいトランザクションをします。 `START TRANSACTION`または`BEGIN WORK`をすることにより、 `AUTOCOMMIT` *false*にするはありません。それはにになります。

これでトランザクションをできます。のなJDBCトランザクションのをしてください。

```
package jdbcTest;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;


public class accTrans {

    public static void doTransfer(double transAmount,int customerIdFrom,int customerIdTo) {

        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        try {
            String DB_CONNECTION_URL =
"jdbc:mysql://localhost:3306/testDB?useUnicode=true&amp;characterEncoding=utf8";
```

```
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection(DB_CONNECTION_URL,DB_USER,USER_PASSWORD);

            --->set auto commit to false
            con.setAutoCommit(false);
            ---> or use con.START TRANSACTION / con.BEGIN WORK

            --->Start SQL Statements for transaction
            --->Checking availability of amount
            double availableAmt    = 0;
            pstmt = con.prepareStatement("SELECT ledgerAmt FROM accTable WHERE customerId=?
FOR UPDATE");
            pstmt.setInt(1, customerIdFrom);
            rs = pstmt.executeQuery();
            if(rs.next())
                availableAmt    = rs.getDouble(1);

            if(availableAmt >= transAmount)
            {
                ---> Do Transfer
                ---> taking amount from cutomerIdFrom
                pstmt = con.prepareStatement("UPDATE accTable SET ledgerAmt=ledgerAmt-? WHERE
customerId=?");
                pstmt.setDouble(1, transAmount);
                pstmt.setInt(2, customerIdFrom);
                pstmt.executeUpdate();

                ---> depositing amount in cutomerIdTo
                pstmt = con.prepareStatement("UPDATE accTable SET ledgerAmt=ledgerAmt+? WHERE
customerId=?");
                pstmt.setDouble(1, transAmount);
                pstmt.setInt(2, customerIdTo);
                pstmt.executeUpdate();

                con.commit();
            }
            --->If you performed any insert,update or delete operations before
            ----> this availability check, then include this else part
            /*else { --->Rollback the transaction if availability is less than required
                con.rollback();
            }*/

        } catch (SQLException ex) {
            ---> Rollback the transaction in case of any error
            con.rollback();
        } finally {
            try {
                if(rs != null)  rs.close();
                if(pstmt != null) pstmt.close();
                if(con != null) con.close();
            }
        }
    }

    public static void main(String[] args) {
        doTransfer(500, 1020, 1021);
        -->doTransfer(transAmount, customerIdFrom, customerIdTo);
    }
}
```

JDBCトランザクションは、トランザクションブロックのSQLのいずれかがした、トランザクションブロックのすべてのSQLがにされたことをし、トランザクションブロックのすべてをし、ロールバックします。

オンラインでトランザクションをむ https://riptutorial.com/ja/mysql/topic/5771/トランザクション

45: トリガー

- CREATE [DEFINER = {ユーザー| CURRENT_USER}] TRIGGER trigger_name trigger_time trigger_eventのtbl_nameをします[trigger_order] trigger_body
- trigger_time{BEFORE |に}
- trigger_event{INSERT || DELETE}
- trigger_order{FOLLOWS | PRECEDES} other_trigger_name

にのトリガーをしているは、2つのポイントでをくがあります。

ごとに

FOR EACH ROWはのです

あなたはのOracleのようにクエリによる トリガーをすることはできません。のけているよりもパフォーマンスにするです

トリガーのまたは

CREATE OR REPLACEは**MySQL**でサポートされていません

MySQLはこのをしていませんが、わりにをします。

```
DELIMITER $$

DROP TRIGGER IF EXISTS myTrigger;
$$
CREATE TRIGGER myTrigger
-- ...

$$
DELIMITER ;
```

これはアトミックトランザクションではないことにしてください。

- `CREATE`した、いトリガーをいます
- いでは、 `DROP`と`CREATE`でのがするがあります`LOCK TABLES myTable WRITE;`して`LOCK TABLES myTable WRITE;`にデータのと`UNLOCK TABLES;`をする`UNLOCK TABLES;`テーブルをする`CREATE`

Examples

なトリガー

テーブルの

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)
```

トリガーの

```
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.06 sec)
```

CREATE TRIGGERステートメントは、アカウントテーブルにけられたins_sumというトリガーをします。また、トリガの、トリガイベント、およびトリガがアクティブになったときのをするもまれます

を

トリガをするには、アキュムレータ@sumをゼロにし、INSERTをしてから、がでどのようなをつかをします。

```
mysql> SET @sum = 0;
mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-100.00);
mysql> SELECT @sum AS 'Total amount inserted';
+-----------------------+
| Total amount inserted |
+-----------------------+
| 1852.48               |
+-----------------------+
```

この、INSERTがされたの@sumのは、14.98 + 1937.50 - 100または1852.48です。

ドロップトリガー

```
mysql> DROP TRIGGER test.ins_sum;
```

をドロップすると、のトリガーもドロップされます。

トリガーの

タイミング

トリガアクションは2つあります。

- BEFOREトリガーがをするにアクティブになり、
- AFTERのトリガー。

トリガーイベント

トリガーをアタッチできるイベントは3つあります。

- INSERT
- UPDATE
- DELETE

のトリガーの

```
DELIMITER $$

CREATE TRIGGER insert_date
    BEFORE INSERT ON stack
    FOR EACH ROW
BEGIN
        -- set the insert_date field in the request before the insert
    SET NEW.insert_date = NOW();
END;

$$
DELIMITER ;
```

のトリガの

```
DELIMITER $$

CREATE TRIGGER update_date
    BEFORE UPDATE ON stack
    FOR EACH ROW
BEGIN
        -- set the update_date field in the request before the update
    SET NEW.update_date = NOW();
END;

$$
DELIMITER ;
```

のトリガの

```
DELIMITER $$

CREATE TRIGGER deletion_date
    AFTER DELETE ON stack
    FOR EACH ROW
BEGIN
        -- add a log entry after a successful delete
    INSERT INTO log_action(stack_id, deleted_date) VALUES(OLD.id, NOW());
```

```
END;

$$
DELIMITER ;
```

オンラインでトリガーをむ https://riptutorial.com/ja/mysql/topic/3069/トリガー

46: ドロップテーブル

- DROP TABLE table_name;
- DROP TABLE IF EXISTSテーブル; - スクリプトのなエラーをけるため
- DROP TABLE t1、t2、t3; - のテーブルをする
- DROP TEMPORARY TABLE t; - CREATE TEMPORARY TABLEからテーブルをする...

パラメーター

パラメーター	
	オプション。これは、DROP TABLEステートメントによってのみをドロップするようにします。
IF	オプション。されている、テーブルの1つがしない、DROP TABLEステートメントはエラーをしません。

Examples

ドロップテーブル

ドロップテーブルは、データベースからテーブルをするためにされます。

テーブルの

tblというのテーブルをし、したテーブルをする

```
CREATE TABLE tbl(
    id INT NOT NULL AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(40) NOT NULL,
    submission_date DATE,
    PRIMARY KEY (id)
);
```

ドロップテーブル

```
DROP TABLE tbl;
```

ごください

テーブルをすると、データベースとそのすべてのがにされ、されません。

データベースからテーブルをする

```
DROP TABLE Database.table_name
```

オンラインでドロップテーブルをむ https://riptutorial.com/ja/mysql/topic/4123/ドロップテーブル

47: ヌル

Examples

NULLの

- `end_date`、 `rating`など、のデータ
- オプションのデータ - たとえば`middle_initial` のとしてはいかもしれませんが
- 0/0 - ゼロをゼロでったようなのの。
- NULLは ""または0のとしくありません。
- その

NULLをテストする

- `IS NULL` / `IS NOT NULL` - `= NULL`はどおりにしません。
- `x <=> y`は "null"です。

`LEFT JOIN`のののののするがしない `a` `b`。

```
SELECT ...
    FROM a
    LEFT JOIN b ON ...
    WHERE b.id IS NULL
```

オンラインでヌルをむ https://riptutorial.com/ja/mysql/topic/6757/ヌル

48: パーティショニング

- **RANGE**パーティショニング 。このタイプのパーティショニングは、されたのについてをパーティションにりてます。
- **LIST**パーティショニング 。 RANGEによるパーティショニングとですが、パーティションがのセットの1つにするについてされるがなります。
- **HASH**パーティショニング 。このタイプのパーティションでは、テーブルにされるのをするユーザーによってされたについてパーティションがされます。このは、でないをするMySQLでなのでできます。このタイプのである`LINEAR HASH`もできます。
- **KEY**パーティショニング 。このタイプのパーティショニングは、されるべき1つのカラムだけがされ、MySQLサーバがのハッシュをするをいて、HASHによるパーティショニングにています。これらのにはのをれることができます。これは、MySQLによってされるハッシュがのデータになくをするためです。このタイプの`LINEAR KEY`もできます。

Examples

RANGEパーティショニング

でられたは、にりのがのにあるをむようにられます。はしているがありますがはなく、 `VALUES LESS THAN`をしてされています。のいくつかのでは、のようなテーブルをして、1から20までの20のビデオストアチェーンのをするとします。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
);
```

このは、にじてさまざまなでごとにパーティションできます。 1つのは、 `store_id`をすることです。たとえば、のように`PARTITION BY RANGE`をして、4つのでをすることができます。

```
ALTER TABLE employees PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

`MAXVALUE`は、なりきなよりもきいをしますにえば、のとしてされます。

MySQLのドキュメントにづいています 。

LISTパーティショニング

リスト·パーティションは、くのでレンジ·パーティションにています。 RANGEによるパーティショニングのとに、パーティションをにするがあります。 2つのタイプのパーティションのないは、リスト·パーティションでは、パーティションが、のしたのセットではなく、リストのセットの1つにあるのメンバーシップにづいておよびされることです。。これはしてわれ`PARTITION BY LIST(expr) expr`のまたはのにづいてされ、をし、そのによってパーティションをする`VALUES IN (value_list)`、 `value_list` Aですカンマでられたのリスト。

のでは、パーティションするテーブルのなは、ここにす`CREATE TABLE`ステートメントによってされるものとします。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
);
```

のにすように、4つのフランチャイズに20のビデオストアがしているとします。

	ID
	3,5,6,9,17
	1,2,10,11,19,20
	4,12,13,14,18
	7,8,15,16

じにするストアのがじパーティションにされるようにこのをパーティションするには

```
ALTER TABLE employees PARTITION BY LIST(store_id) (
    PARTITION pNorth VALUES IN (3,5,6,9,17),
    PARTITION pEast VALUES IN (1,2,10,11,19,20),
    PARTITION pWest VALUES IN (4,12,13,14,18),
    PARTITION pCentral VALUES IN (7,8,15,16)
);
```

MySQLのドキュメントにづいています 。

ハッシュパーティショニング

HASHによるパーティショニングは、に、ののパーティションでのデータのなをするためにされます。またはリスト·パーティションでは、されたまたはのセットがされるパーティションをにするがあります。ハッシュパーティショニングでは、MySQLがこれをし、ハッシュされるのと、されたテーブルがされるパーティションのにづいて、のまたはをするだけでみます。

のは、store_idのハッシングをするテーブルをし、4つのパーティションにします。

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY HASH(store_id)
PARTITIONS 4;
```

`PARTITIONS`をしないと、パーティションのはデフォルトで1になります。

MySQLのドキュメントにづいています 。

オンラインでパーティショニングをむ https://riptutorial.com/ja/mysql/topic/5128/パーティショニング

49: パスワードをする

Examples

LinuxでMySQLルートパスワードをする

MySQLのrootユーザのパスワードをするには

ステップ1 MySQLサーバをします。

- UbuntuまたはDebianで
  `sudo /etc/init.d/mysql stop`
- CentOS、Fedora、またはRed Hat Enterprise Linuxの
  `sudo /etc/init.d/mysqld stop`

ステップ2システムなしでMySQLサーバをします。

```
sudo mysqld_safe --skip-grant-tables &
```

`mysqld_safe`ができないは、

```
sudo mysqld --skip-grant-tables &
```

ステップ3 MySQLサーバにします。

```
mysql -u root
```

ステップ4 rootユーザのしいパスワードをします。

5.7

```
FLUSH PRIVILEGES;
ALTER USER 'root'@'localhost' IDENTIFIED BY 'new_password';
FLUSH PRIVILEGES;
exit;
```

5.7

```
FLUSH PRIVILEGES;
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
FLUSH PRIVILEGES;
exit;
```

`ALTER USER`はMySQL 5.7.6でされました。

ステップ5 MySQLサーバをします。

- UbuntuまたはDebianで
  ```
  sudo /etc/init.d/mysql stop
  sudo /etc/init.d/mysql start
  ```
- CentOS、Fedora、またはRed Hat Enterprise Linuxの
  ```
  sudo /etc/init.d/mysqld stop
  sudo /etc/init.d/mysqld start
  ```

WindowsでのMySQLルートパスワードの

Windowsでrootのパスワードをするはのをするがあります。

1のいずれかのをしてコマンドプロンプトをします。

Perst Crtl `Crtl+R` or Go `Start Menu > Run`して`cmd`としてEnterキーをす

ステップ2あなたのディレクトリを`MYSQL`がインストールされているにします。

```
C:\> cd C:\mysql\bin
```

ステップ3は`mysql`コマンドプロンプトをするがあります

```
C:\mysql\bin> mysql -u root mysql
```

ステップ4クエリをして`root`パスワードをする

```
mysql> SET PASSWORD FOR root@localhost=PASSWORD('my_new_password');
```

プロセス

1. MySQLmysqldサーバ/デーモンプロセスをします。
2. MySQLサーバをして、--skip-grant-tablesオプションをして、パスワードをしないようにします `mysqld_safe --skip-grant-tables &`
3. rootユーザとしてMySQLサーバにします `mysql -u root`
4. パスワードをする

   - 5.7.6 `ALTER USER 'root'@'localhost' IDENTIFIED BY 'new-password';`
   - 5.7.5のMariaDB `SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new-password); flush privileges; quit;`

5. MySQLサーバをします。

これはにじサーバーにいるにのみします。

オンラインドキュメント http : //dev.mysql.com/doc/refman/5.7/en/resetting-permissions.html

オンラインでパスワードをするをむ https://riptutorial.com/ja/mysql/topic/2761/パスワードをする

50: バックティック

Examples

バックティックスの

そこバッククォートは、クエリでされているくのがありますが、くのにとって、それはバッククォートをすることはとしてとき、またはどこです。。

バッククォートは、に「 *MySQL* 」とばれるエラーをぐためにされます 。 PHPmyAdminでテーブルをするときに、「 *MySQL* 」をしているというまたはがされることがあります 。

たとえば、 "`group`"というのをつテーブルをすると、がされます。これは、のせをうことができるためです。

```
SELECT student_name, AVG(test_score) FROM student GROUP BY group
```

クエリでエラーがしないようにするには、バッククォートをしてクエリをするがあります。

```
SELECT student_name, AVG(test_score) FROM student GROUP BY `group`
```

カラムは、バッククォートでむことができるだけでなく、テーブルでむこともできます。たとえば、のテーブルを`JOIN`するがあるなどです。

```
SELECT `users`.`username`, `groups`.`group` FROM `users`
```

みやすい

テーブルとカラムのにバッククォートをすると、クエリをみやすくなります。

たとえば、クエリーをすべてでくには、のようにします。

```
select student_name, AVG(test_score) from student group by group
select `student_name`, AVG(`test_score`) from `student` group by `group`
```

MySQLのマニュアルページのキーワードとをしてください。 Rがいているものはです。はにキーワードです。リザーブドにはながです。

オンラインでバックティックをむ https://riptutorial.com/ja/mysql/topic/5208/バックティック

51: ピボットクエリ

MySQLのピボットクエリのは、 `GROUP_CONCAT()`にしています。ピボットクエリのをするのがきいとされるは、 `group_concat_max_len`のを`group_concat_max_len`するがあります。

```
set session group_concat_max_len = 1024 * 1024; -- This should be enough for most cases
```

Examples

ピボットクエリの

MySQLは、ピボットクエリをするためのみみのをしていません。ただし、これらはプリペアドステートメントをしてできます。

テーブル`tbl_values`します。

イド		グループ	
1	ピート	A	10
2	ピート	B	20
3	ジョン	A	10

リクエスト`Name`の`Value`のをすクエリをします。 `Group`はヘッダーでなければならず、 `Name`はヘッダーでなければなりません。

```
-- 1. Create an expression that builds the columns
set @sql = (
    select group_concat(distinct
        concat(
            "sum(case when `Group`='", Group, "' then `Value` end) as `", `Group`, "`"
        )
    )
    from tbl_values
);

-- 2. Complete the SQL instruction
set @sql = concat("select Name, ", @sql, " from tbl_values group by `Name`");

-- 3. Create a prepared statement
prepare stmt from @sql;

-- 4. Execute the prepared statement
execute stmt;
```

	A	B
ジョン	10	ヌル
ピート	10	20

になったみのステートメントのりてをする

```
deallocate prepare stmt;
```

SQL Fiddleの

オンラインでピボットクエリをむ https://riptutorial.com/ja/mysql/topic/3074/ピボットクエリ

52: リミットとオフセット

- SELECT column_1 [、column_2]
  FROM table_1
  ORDER BY order_column
  LIMIT row_count [OFFSET row_offset]
- SELECT column_1 [、column_2]
  FROM table_1
  ORDER BY order_column
  LIMIT [row_offset、] row_count

「」は「テーブルのの」をします。

「オフセット」rowからのピックをしますプライマリキーまたはフィールドデータとしないでください

Examples

リミットとオフセットの

のusersテーブルをしてください。

id	ユーザー
1	ユーザー1
2	User2
3	ユーザー3
4	ユーザー4
5	User5

SELECTせのセットのをするために、 LIMITを1つまたは2つののとともにとしてできますゼロをむ。

1つのをつLIMIT

1つのをすると、セットはのようにされたにされます。

```
SELECT * FROM users ORDER BY id ASC LIMIT 2
```

id	ユーザー
1	ユーザー1
2	User2

のが0、セットはになります。

また、されるセットののをするためには、 `ORDER BY`があることにしてくださいのでべえる。

2つのをつ`LIMIT`

`LIMIT`で2つのをするは、のようになります。

- のは、セットのがされるをします。このは、のあるセットのののののをすため、 オフセット とばれることがよくあります。これにより、はとして0をけることができ、きセットののをにれることができます。
- **2**のはセットにされるのをします1つののと。

したがって、クエリ

```
SELECT * FROM users ORDER BY id ASC LIMIT 2, 3
```

のセットをします。

id	ユーザー
3	ユーザー3
4	ユーザー4
5	User5

offsetが0、セットは1つの`LIMIT`にすることにしてください。これは、の2つのクエリをします。

```
SELECT * FROM users ORDER BY id ASC LIMIT 0, 2

SELECT * FROM users ORDER BY id ASC LIMIT 2
```

じセットをする

id	ユーザー
1	ユーザー1
2	User2

OFFSET キーワード

2つのをつLIMITののは、のようにののにOFFSETキーワードをすることです。

```
SELECT * FROM users ORDER BY id ASC LIMIT 2 OFFSET 3
```

このクエリは、のセットをします。

id	ユーザー
3	ユーザー3
4	ユーザー4

このでは、のがりえられていることにしてください。

- のはセットでされるをします。
- **2**のはオフセットをします。

オンラインでリミットとオフセットをむ https://riptutorial.com/ja/mysql/topic/548/リミットとオフセット

53: ログファイル

Examples

リスト

- なログ - すべてのクエリ - VARIABLE general_logを
- いログ - long_query_timeよりもいクエリ - slow_query_log_file
- Binlog - レプリケーションとバックアップ - log_bin_basename
- リレーログ - レプリケーション
- なエラー - mysqld.err
- / - mysql.logそれほどくない - log_error
- InnoDB REDOログ - iblog *

くのログのデフォルトのは、`basedir`と`datadir`をしてください

いくつかのログは、のによってオン/オフされます。いくつかはファイルまたはテーブルにきまれます。

へのこれはとながです

Documenters Windowsと* nixのについて、ログタイプのデフォルトのとをめてください。 またはできるだけくの

クエリログ

スロー·クエリー·ログは、 `long_query_time`を`long_query_time`てするクエリーのログ·イベントでされます。えば、までに10。されているのしきいをするには、のコマンドをします。

```
SELECT @@long_query_time;
+-------------------+
| @@long_query_time |
+-------------------+
|         10.000000 |
+-------------------+
```

これは、 `my.cnf`または`my.ini`ファイル`my.cnf` GLOBALとしてできます。または、これはによってできますが、これはしいことです。は010のでできます。どのようなをする

- 10はににたないほどいです。
- 2はである。
- 0.5およびのがである。
- 0はすべてをキャプチャします。これはなほどにディスクをいっぱいにするがありますが、にです。

せのは、オンまたはオフのいずれかでわれます。また、ログにされたファイルもされます。に、これらのをりげます。

```
SELECT @@slow_query_log; -- Is capture currently active? (1=On, 0=Off)
SELECT @@slow_query_log_file; -- filename for capture. Resides in datadir
SELECT @@datadir; -- to see current value of the location for capture file

SET GLOBAL slow_query_log=0; -- Turn Off
-- make a backup of the Slow Query Log capture file. Then delete it.
SET GLOBAL slow_query_log=1; -- Turn it back On (new empty file is created)
```

については、MySQLのマニュアルペ―ジのSlow Query Logをしてください。

slowlogのオン/オフにするのは、5.6でされました。いバ―ジョンにはのみがありました。

があなたのシステムをくしているかをるための「の」

```
long_query_time=...
turn on the slowlog
run for a few hours
turn off the slowlog (or raise the cutoff)
run pt-query-digest to find the 'worst' couple of queries.  Or mysqldumpslow -s t
```

なクエリログ

General Query Logには、クライアントの、、およびクエリからのなのリストがまれています。これはデバッグにとってにですが、それはパフォ―マンスのげになります。

なクエリログのをにします。

```
36 Query insert questions_c23(qId,ownerId,title,votes,answers,isClosed,closeVotes,views,owne
        comments,answeredAccepted,askDate,closeDate,lastScanDate,ign,bn,pvtc,
        mainTagForImport,prepStatus,touches,status,status_bef_change,cv_bef_change,max_cv_r
        values(38666373, 1322183, 'How to post a numeric value in c#', 0, 1, 0, 0, 50, 1,
        0, 0, '2016-07-29 19:40:32', null, now(), 0, 0, 0,
        'c%23',0,1,'O','',0,0)
        on duplicate key update title='How to post a numeric value in c#', votes=0, answers
        answeredAccepted=0,lastScanDate=now(), touches=touches+1,status='O'
```

ログがキャプチャされているかどうかをするには

```
SELECT @@general_log; -- 1 = Capture is active; 0 = It is not.
```

キャプチャファイルのファイルをするには

```
SELECT @@general_log_file; -- Full path to capture file
```

ファイルのフルパスがされない、ファイルがするで`datadir`。

Windowsの

```
+-----------------------------------------------------------+
| @@general_log_file                                        |
+-----------------------------------------------------------+
| C:\ProgramData\MySQL\MySQL Server 5.7\Data\GuySmiley.log |
+-----------------------------------------------------------+
```

Linux

```
+-----------------------------------+
| @@general_log_file                |
+-----------------------------------+
| /var/lib/mysql/ip-ww-xx-yy-zz.log |
+-----------------------------------+
```

`general_log_file` GLOBALがされると、しいログが`datadir`されます。ただし、をべることでフルパスがされなくなることがあります。

ファイルの`general_log_file`にエントリがない、デフォルトでは`datadir` `@@hostname` .logになり`datadir`。

キャプチャをオフにするのがベストプラクティスです。キャプチャのとをしたファイルをつバックアップディレクトリにログファイルをします。のファイルをするのは、そのファイルのファイルシステムのがしていないです。ログファイルのしいファイルをし、キャプチャをオンにします。ベストプラクティスには、でキャプチャしたいでもながまれます。、キャプチャはデバッグのでのみオンになっています。

バックアップされたログのなファイルシステムファイルはのようになります。

```
/LogBackup/GeneralLog_20160802_1520_to_20160802_1815.log
```

ここで、とはとしてのファイルのです。

Windowsの、をしてのにしてください。

```
SELECT @@general_log; -- 0. Not being captured
SELECT @@general_log_file; -- C:\ProgramData\MySQL\MySQL Server 5.6\Data\GuySmiley.log
SELECT @@datadir; -- C:\ProgramData\MySQL\MySQL Server 5.7\Data\
SET GLOBAL general_log_file='GeneralLogBegin_20160803_1420.log'; -- datetime clue
SET GLOBAL general_log=1; -- Turns on actual log capture. File is created under `datadir`
SET GLOBAL general_log=0; -- Turn logging off
```

Linuxもです。これらはなをします。サ―バをすると、ファイルのがされます。

ファイルについては、のするのをしてください。

```
[mysqld]
general_log_file = /path/to/currentquery.log
general_log      = 1
```

さらに、`log_output`は、 `FILE`だけでなく、 `TABLE`にすることもできます。そのために、 をごくださ

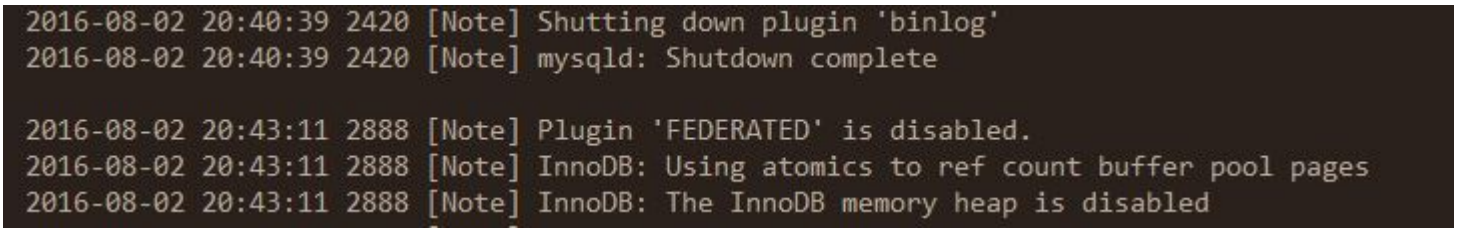
い。

MySQLのマニュアルページのクエリログをしてください。

エラーログ

エラー・ログには、および、およびサーバーがしたなイベントがされます。

そののはのとおりです。



`log_error`は、エラー・ロギングのためのログ・ファイルへのパスがされます。

`log_error`ファイルエントリがない、システムはそのを`datadir @@hostname datadir`デフォルトします。 `log_error`はではないことにしてください。したがって、はcnfまたはiniファイルのとサーバーのまたは、ここではマニュアルページのリンクの「エラーログファイルのフラッシュとの」をによってわれます。

エラーの、ロギングをにすることはできません。をトラブルシューティングするにはシステムのにとってです。また、エントリはクエリログとしてまれです。

GLOBAL`log_warnings`は、サーバのバージョンによってなるレベルをします。のスニペットでは、

```
SELECT @@log_warnings; -- make a note of your prior setting
SET GLOBAL log_warnings=2; -- setting above 1 increases output (see server version)
```

の`log_warnings`はです。

`cnf`ファイルと`ini`ファイルのファイルのは、のようになります。

```
[mysqld]
log_error        = /path/to/CurrentError.log
log_warnings     = 2
```

MySQL 5.7.2では、レベルのを3にし、GLOBAL `log_error_verbosity`をしました。び、それは5.7.2でされました。にしたり、としてチェックしたり、 `cnf`や`ini`ファイルのでチェックすることができます。

MySQL 5.7.2

```
[mysqld]
log_error           = /path/to/CurrentError.log
log_warnings        = 2
```

```
log_error_verbosity = 3
```

しMySQLのマニュアルペ―ジをしてくださいエラ―ログ 、にフラッシングのためにと、エラ―ログファイルのの、およびにするバ―ジョンとのエラ―ログセクション `log_warnings`と `error_log_verbosity`。

オンラインでログファイルをむ https://riptutorial.com/ja/mysql/topic/5102/ログファイル

54: テーブル

Examples

テーブルの

テンポラリテーブルは、なデータをするのににです。 Temporary tablesオプションは、MySQLバージョン3.23でできます。

テンポラリテーブルは、セッションがするかがじられるとにされます。ユーザーはをドロップすることもできます。

テンポラリ・テーブルは、そのテーブルをしたクライアントだけがでアクセスであるため、にじテンポラリ・テーブルをくのでできます。

テーブルは、のタイプでできます

```
--->Basic temporary table creation
CREATE TEMPORARY TABLE tempTable1(
       id INT NOT NULL AUTO_INCREMENT,
       title VARCHAR(100) NOT NULL,
       PRIMARY KEY ( id )
    );

--->Temporary table creation from select query
CREATE TEMPORARY TABLE tempTable1
    SELECT ColumnName1,ColumnName2,... FROM table1;
```

テーブルをするときにインデックスをできます。

```
CREATE TEMPORARY TABLE tempTable1
        ( PRIMARY KEY(ColumnName2) )
    SELECT ColumnName1,ColumnName2,... FROM table1;
```

`IF NOT EXISTS` *'table already exists'*というエラーをけるために、のようにキーワードをすることができます。ただし、しているテーブルがのセッションにすでにするは、テーブルはされません。

```
CREATE TEMPORARY TABLE IF NOT EXISTS tempTable1
    SELECT ColumnName1,ColumnName2,... FROM table1;
```

テーブルの

ドロップテーブルは、のセッションでしたテーブルをするためにします。

```
DROP TEMPORARY TABLE tempTable1

DROP TEMPORARY TABLE IF EXISTS tempTable1
```

`IF EXISTS`をすると、しないのあるテーブルでエラーがしないようにすることができます

オンラインでテーブルをむ https://riptutorial.com/ja/mysql/topic/5757/テーブル

55:

き

MySQLにはというながあります 。は、テーブルやなどのとしてバッククォート `でまれているにのみできます。それのは、エラーがします。

このようなエラーをするには、をとしてしないでください。

すべてのはのとおりです ドキュメント から

- アクセスな
-
- すべて
- ALTER
-
- そして
- として
- ASC
- な
-
- のに
- BIGINT
- バイナリ
- BLOB
-
- によって
- コール
- カスケード
-
- する
- CHAR
- キャラクター
- チェック
- COLLATE
- カラム
-
- コンストレイント
- する
- コンバート
- CREATE
- クロス
- の
- の

- CURRENT_TIMESTAMP
- の
- カーソル
- データベース
- データベース
- DAY_HOUR
- DAY_MICROSECOND
- DAY_MINUTE
- DAY_SECOND
- DEC
- DECIMAL
-
- デフォルト
-
-
- DESC
- DESCRIBE
-
- DISTINCT
- DISTINCTROW
- DIV
- ダブル
- ドロップ
- デュアル
-
- ELSE
- ELSEIF
-
- エスケープ
-
-
- する
-
- フェッチ
- く
- FLOAT4
- FLOAT8
- にとって
-
-
- から
- フルテックス
- GENERATED
- する
-
- グループ
- HAVING

- HIGH_PRIORITY
- HOUR_MICROSECOND
- HOUR_MINUTE
- HOUR_SECOND
- IF
- IGNORE
- に
- INDEX
- INFILE
- インナー
- INOUT
-
- インサート
- INT
- INT1
- INT2
- INT3
- INT4
- INT8
-
-
- に
- IO_AFTER_GTIDS
- IO_BEFORE_GTIDS
- IS
- ITERATE
- ジョイン
- キー
- キーズ
- します
-
- れる
-
- き
-
- リニア
- ライン
-
-
- LOCALTIMESTAMP
- ロック
- いです
- LONGBLOB
- ロングテックス
- ループ
- LOW_PRIORITY

- MASTER_BIND
- MASTER_SSL_VERIFY_SERVER_CERT
-
- MAXVALUE
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXT
- ミドル
- MINUTE_MICROSECOND
- MINUTE_SECOND
- MOD
-
- ナチュラル
- NOT
- NO_WRITE_TO_BINLOG
- ヌル
-
- に
-
- OPTIMIZER_COSTS
- オプション
- OPTIONALLY
- または
-
- でる
-
- OUTFILE
- パーティション
-
-
-
- パージ
-
- む
- READS
- みき
- リアル
-
- REGEXP
- リリース
- リネーム
- りす
-
-
-
-
- り
-

りす
-
- RLIKE
- SCHEMA
- シェーマス
- SECOND_MICROSECOND
- セレクト
-
- セパレータ
- セット
- ショー
-
- SMALLINT
- スパイラル
-
- SQL
- SQLEXCEPTION
- SQLSTATE
- SQLWARNING
- SQL_BIG_RESULT
- SQL_CALC_FOUND_ROWS
- SQL_SMALL_RESULT
- SSL
-
- ストアド
- STRAIGHT_JOIN
-
- した
- その
- TINYBLOB
- TINYINT
- TINYTEXT
- に
- TRAILING
- き
-
- UNDO
-
- ユニーク
- UNLOCK
- UNSIGNED
-
-
- つかいます
- USING
- UTC_DATE
- UTC_TIME

- UTC_TIMESTAMP
- VALUES
-
- VARCHAR
- VARCHARACTER
- バリエーション
- バーチャル
- いつ
- どこに
- もなく
- WITH
- きます
- XOR
-
- ゼロフィル
- GENERATED
- OPTIMIZER_COSTS
- ストアド
- バーチャル

Examples

によるエラー

このような`order`というテーブルからしようとすると

```
select * from order
```

エラーがします。

> エラーコード1064. SQLにエラーがあります。あなたのMySQLサーバのバージョンにするマニュアルをチェックし、しいが1の 'order'のくでされるようにしてください

MySQLのみのキーワードはバッククォート、でエスケープするがあります

```
select * from `order`
```

キーワードとまたはをすることができます。

MySQLでをテーブルまたはカラムとしてするためのエラー 。

オンラインでをむ https://riptutorial.com/ja/mysql/topic/1398/

56: の

- ALTER [ ] TABLE tbl_name [**alter_specification** [、alter_specification] ...] [partition_options]

```
    alter_specification: table_options
      | ADD [COLUMN] col_name column_definition [FIRST | AFTER col_name ]
      | ADD [COLUMN] (col_name column_definition,...)
      | ADD {INDEX|KEY} [index_name] [index_type] (index_col_name,...) [index_option] ...
      | ADD [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...) [index_option]
...
      | ADD [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] [index_type]
(index_col_name,...) [index_option] ...
      | ADD FULLTEXT [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
      | ADD SPATIAL [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
      | ADD [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...)
reference_definition
      | ALGORITHM [=] {DEFAULT|INPLACE|COPY}
      | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
      | CHANGE [COLUMN] old_col_name new_col_name column_definition [FIRST|AFTER col_name]
      | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
      | MODIFY [COLUMN] col_name column_definition [FIRST | AFTER col_name]
      | DROP [COLUMN] col_name
      | DROP PRIMARY KEY
      | DROP {INDEX|KEY} index_name
      | DROP FOREIGN KEY fk_symbol
      | DISABLE KEYS
      | ENABLE KEYS
      | RENAME [TO|AS] new_tbl_name
      | RENAME {INDEX|KEY} old_index_name TO new_index_name
      | ORDER BY col_name [, col_name] ...
      | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
      | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
      | DISCARD TABLESPACE
      | IMPORT TABLESPACE
      | FORCE
      | {WITHOUT|WITH} VALIDATION
      | ADD PARTITION (partition_definition)
      | DROP PARTITION partition_names
      | DISCARD PARTITION {partition_names | ALL} TABLESPACE
      | IMPORT PARTITION {partition_names | ALL} TABLESPACE
      | TRUNCATE PARTITION {partition_names | ALL}
      | COALESCE PARTITION number
      | REORGANIZE PARTITION partition_names INTO (partition_definitions)
      | EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH|WITHOUT} VALIDATION]
      | ANALYZE PARTITION {partition_names | ALL}
      | CHECK PARTITION {partition_names | ALL}
      | OPTIMIZE PARTITION {partition_names | ALL}
      | REBUILD PARTITION {partition_names | ALL}
      | REPAIR PARTITION {partition_names | ALL}
      | REMOVE PARTITIONING
      | UPGRADE PARTITIONING
   index_col_name: col_name [(length)] [ASC | DESC]
   index_type: USING {BTREE | HASH}
   index_option: KEY_BLOCK_SIZE [=] value
      | index_type
      | WITH PARSER parser_name
```

```
      | COMMENT 'string'
```

table_options: table_option [[,] table_option] ... (see CREATE TABLE options) table_options: table_option [[,] table_option] ... (see options)

partition_options: (see CREATE TABLE options) partition_options: (see options)

リファレンス MySQL 5.7リファレンスマニュアル/ ... / ALTER TABLE/ 14.1.8 ALTER TABLE

Examples

ストレージエンジンのテーブルをする。するfile_per_table

たとえば、 t1がInnoDBテーブルでない、このステートメントはストレージエンジンをInnoDBにします。

```
ALTER TABLE t1 ENGINE = InnoDB;
```

テーブルがすでにInnoDBのは、テーブルとそのインデックスがされ、 OPTIMIZE TABLEとのがあり OPTIMIZE TABLE。 ディスクスペースをしでもやすことができます。

innodb_file_per_tableのがt1ビルドになとなる、これはfile_per_tableにされますまたは file_per_tableからされます。

テーブルのALTER COLUMN

```
CREATE DATABASE stackoverflow;

USE stackoverflow;

Create table stack(
    id_user int NOT NULL,
    username varchar(30) NOT NULL,
    password varchar(30) NOT NULL
);

ALTER TABLE stack ADD COLUMN submit date NOT NULL; -- add new column
ALTER TABLE stack DROP COLUMN submit; -- drop column
ALTER TABLE stack MODIFY submit DATETIME NOT NULL; -- modify type column
ALTER TABLE stack CHANGE submit submit_date DATETIME NOT NULL; -- change type and name of
column
ALTER TABLE stack ADD COLUMN mod_id INT NOT NULL AFTER id_user; -- add new column after
existing column
```

ALTERテーブルのINDEX

パフォーマンスをさせるために、にをすることができます

```
ALTER TABLE TABLE_NAME ADD INDEX `index_name` (`column_name`)
```

ィンデックスをするようにする

```
ALTER TABLE TABLE_NAME ADD INDEX `index_name` (`col1`,`col2`)
```

インクリメントをする

インクリメントをすると、のにAUTO_INCREMENTにギャップをれたくないにです。

たとえば、ながテーブルにされ、した、インクリメントのギャップをしたいとします。AUTO_INCREMENTのMAXが100であるとします。オートインクリメントをするには、のようにします。

```
ALTER TABLE your_table_name AUTO_INCREMENT = 101;
```

キーのタイプの

```
ALTER TABLE fish_data.fish DROP PRIMARY KEY;
ALTER TABLE fish_data.fish MODIFY COLUMN fish_id DECIMAL(20,0) NOT NULL PRIMARY KEY;
```

キーをせずにこののタイプをしようとすると、エラーがします。

をする

dbのをすると、たとえばのクエリをできます。このdbスキーマがある

```
users (
    firstname varchar(20),
    lastname varchar(20),
    age char(2)
)
```

`age`のを`char`から`int`にするには、のクエリをします。

```
ALTER TABLE users CHANGE age age tinyint UNSIGNED NOT NULL;
```

なフォーマットはのとおりです。

```
ALTER TABLE table_name CHANGE column_name new_column_definition
```

MySQLデータベースのをする

MySQLデータベースのをするコマンドは1つではありませんが、なをしてバックアップとリストアをうことでこれをできます。

```
mysqladmin -uroot -p<password> create <new name>
mysqldump -uroot -p<password> --routines <old name> | mysql -uroot -pmypassword <new name>
```

```
mysqladmin -uroot -p<password> drop <old name>
```

ステップ

1. のをテキストエディタにコピーします。
2. すると<old name>、 <new name>および<password> のユーザーをするはオプションで<root>へのすべてのをきえます。
3. コマンドラインで1つずつしますMySQLの "bin"フォルダがパスにあるとし、プロンプトがされたら "y"をします。

1つのデータベースからのテーブルにをします。テーブルでこれをいます

```
RENAME TABLE `<old db>`.`<name>` TO  `<new db>`.`<name>`;
```

これらのをするには、のようにします。

```
SELECT CONCAT('RENAME TABLE old_db.', table_name, ' TO ',
                         'new_db.', table_name)
    FROM information_schema.TABLES
    WHERE table_schema = 'old_db';
```

。ファイルシステムのファイルをにかすだけで、テーブルやデータベースをしないでください。これはMyISAMののところでうまくいきましたが、InnoDBとテーブルスペースのしいにはうまくいきません。に、 "Data Dictionary"がファイルシステムからシステムInnoDBテーブルにされたとき、おそらくのメジャーリリースになります。 InnoDBテーブルの<PARTITION>をに<DROPping>とはにさせるには、 "トランスポータブルテーブルスペース"をするがあります。いには、くファイルもありません。

2つのMySQLデータベースのをする

のコマンドをして、2つのMySQLデータベース <db1>および<db2> のをスワップできます。

```
mysqladmin -uroot -p<password> create swaptemp
mysqldump -uroot -p<password> --routines <db1> | mysql -uroot -p<password> swaptemp
mysqladmin -uroot -p<password> drop <db1>
mysqladmin -uroot -p<password> create <db1>
mysqldump -uroot -p<password> --routines <db2> | mysql -uroot -p<password> <db1>
mysqladmin -uroot -p<password> drop <db2>
mysqladmin -uroot -p<password> create <db2>
mysqldump -uroot -p<password> --routines swaptemp | mysql -uroot -p<password> <db2>
mysqladmin -uroot -p<password> drop swaptemp
```

ステップ

1. のをテキストエディタにコピーします。
2. すると<db1>、 <db2>および<password> のユーザーをするはオプションで<root>へのすべてのをきえます。
3. コマンドラインで1つずつしますMySQLの "bin"フォルダがパスにあるとし、プロンプトが

されたら "y"をします。

MySQLテーブルのをする

1つのコマンドでテーブルのをすることができます

```
RENAME TABLE `<old name>` TO `<new name>`;
```

のはまったくじです

```
ALTER TABLE `<old name>` RENAME TO `<new name>`;
```

テンポラリ·テーブルのをするは、の`ALTER TABLE`バージョンをするがあります。

ステップ

1. のの`<old name>`と`<new name>`をするにきえます。 テーブルがのデータベースにされているは、 *dbname*。 *tablename*は、 *<old name>*および/または*<new name>*できます。
2. MySQLコマンドラインのデータベースまたはMySQL Workbenchなどのクライアントでします。 ユーザーは、いにしては*ALTER*および*DROP*を、しいにしては*CREATE*および*INSERT*をっているがあります。

MySQLテーブルののをする

のは、のでうこともできますが、しい、「」つまり、そのデータとNULL、インクリメントなどのそののオプションプロパティもするがあります。

```
ALTER TABLE `<table name>` CHANGE `<old name>` `<new name>` <column definition>;
```

ステップ

1. MySQLコマンドラインまたはMySQL Workbenchなどのクライアントをきます。
2. のをします`SHOW CREATE TABLE <table name>;` `<table name>`をするにきえる。
3. をするのをきめますつまり、ののにされますが、のからるのすべて 。
4. のの`<old name>` 、 `<new name>`および`<column definition>`をするにきえてします。

オンラインでのをむ https://riptutorial.com/ja/mysql/topic/2627/の

57:

き

MySQLはFULLTEXTをします。これは、とフレーズにもよくマッチするテキストをむをつをします。

FULLTEXTは、のをむではなをします。だから、あなたがそれをしているときに、のテーブルをオンラインでするとにつかもしれません。タイトルとのアイテムのです 。ダウンロードしてし、MySQLにロードすることができます。

FULLTEXTはのをとしています。これは、のWHERE column LIKE 'text%'フィルタリングよりくのをするようにされていWHERE column LIKE 'text%'。

FULLTEXTはMyISAMテーブルでできます。 MySQLバージョン5.6.4のInnoDBテーブルでもできます。

Examples

シンプルなFULLTEXT

```
    SET @searchTerm= 'Database Programming';
SELECT MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) Score,
       ISBN, Author, Title
  FROM book
 WHERE MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE)
 ORDER BY MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) DESC;
```

ISBN 、 'Title'、 'Author'というのをつbookというテーブルがある、これは'Database Programming'というにするをします。それはにのマッチをしています。

これをうには、 Titleのをにするがあります。

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_index (Title);
```

シンプルなBOOLEAN

```
    SET @searchTerm= 'Database Programming -Java';
SELECT MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE) Score,
       ISBN, Author, Title
  FROM book
 WHERE MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE)
 ORDER BY MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE) DESC;
```

ISBN 、 Title 、 Authorというのをつbookというがある、 Titleは'Database'と'Programming'というがまれていますが、 'Java'というはされません。

これをうには、タイトルのをにするがあります。

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_index (Title);
```

FULLTEXT

```
    SET @searchTerm= 'Date Database Programming';
 SELECT MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) Score,
        ISBN, Author, Title
   FROM book
  WHERE MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE)
  ORDER BY MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) DESC;
```

`ISBN`、 `Title`、 `Author`をつbookというがある、これは 'Date Database Programming'というにするをします。それはにのマッチをしています。のには、CJ Dateのがあります。

ただし、ベストマッチの1つは、 デーートののガイドのからなメイトへのです。これはFULLTEXTのをしています。けされたの

これをさせるには、TitleとAuthorのをにするがあります。

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_author_index (Title, Author);
```

オンラインでをむ https://riptutorial.com/ja/mysql/topic/8759/

58:

- DELETE [LOW_PRIORITY] [QUICK] []テーブルから[WHERE] [ORDER BY[ASC | DESC]] [LIMIT number_rows]; ///テーブルからをする

パラメーター

パラメータ	
LOW_PRIORITY	`LOW_PRIORITY`がされていると、テーブルからみったプロセスがなくなるまでがれます
IGNORE	`IGNORE`と、にしたすべてのエラーはされます
	レコードをするテーブル
WHERE	レコードをするためにたすがある。がされていないは、テーブルのすべてのレコードがされます
ORDER BY	`ORDER BY`がされている、レコードはされたでされます
	テーブルからするレコードのをします。えられた`number_rows`がされます。

Examples

Whereでする

```
DELETE FROM `table_name` WHERE `field_one` = 'value_one'
```

これは、そのの`field_one`のが 'value_one'とするテーブルからすべてのをします

`WHERE`はselectとじようにするので、 `>` 、 `<` 、 `<>`や`LIKE`などのものができます。

クエリにはWHERE、LIKEをするがあります。をしない、そののすべてのデータがされます。

テーブルからすべてのをする

```
DELETE FROM table_name ;
```

これにより、テーブルのすべてのがすべてされます。のもなです。また、 `WHERE`がされていると、テーブルをにするがあるため、 `DELETE`をにするがあります。

をする

```
DELETE FROM `table_name` WHERE `field_one` = 'value_one' LIMIT 1
```

これは 'Where with Whereをする'とじでしますが、されたがされるとをします。

このようにするをするは、にするのがされることにしてください。にされていない、はソ―トされずにされるがあるため、りのものではないがあります。

テ―ブルの

MySQLの`DELETE`ステ―トメントは、 `JOIN`をして、するテ―ブルをすることもできます。これはネストされたクエリをけるのにです。えられたスキ―マ

```
create table people
(    id int primary key,
    name varchar(100) not null,
    gender char(1) not null
);
insert people (id,name,gender) values
(1,'Kathy','f'),(2,'John','m'),(3,'Paul','m'),(4,'Kim','f');

create table pets
(    id int auto_increment primary key,
    ownerId int not null,
    name varchar(100) not null,
    color varchar(100) not null
);
insert pets(ownerId,name,color) values
(1,'Rover','beige'),(2,'Bubbles','purple'),(3,'Spot','black and white'),
(1,'Rover2','white');
```

id		
1	キャシ―	f
2	ジョン	m
3	ポ―ル	m
4	キム	f

id	ownerId		
1	1	ロ―バ―	ベ―ジュ
2	2		の
4	1	ロ―バ―2	

ポールのペットをりきたいは、

```
DELETE p2
FROM pets p2
WHERE p2.ownerId in (
    SELECT p1.id
    FROM people p1
    WHERE p1.name = 'Paul');
```

のようにきすことができます。

```
DELETE p2    -- remove only rows from pets
FROM people p1
JOIN pets p2
ON p2.ownerId = p1.id
WHERE p1.name = 'Paul';
```

*1*がされました
スポットはペットからされます

`p1`と`p2`はテーブルのエイリアスです。にいテーブルやみやすさのためにです。

とペットのをするには

```
DELETE p1, p2     -- remove rows from both tables
FROM people p1
JOIN pets p2
ON p2.ownerId = p1.id
WHERE p1.name = 'Paul';
```

*2*がされました
スポットはペットからされます
PaulがPeopleからされました

キー

DELETEがforeingキーきのをむとき、オプティマイザはにわないでをすることがあります。たとえば、 `pets`のにキーをする

```
ALTER TABLE pets ADD CONSTRAINT `fk_pets_2_people` FOREIGN KEY (ownerId) references people(id)
ON DELETE CASCADE;
```

エンジンが`pets`に`people`からエントリをしようとすると、のエラーがするがあります。

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
(`test`.`pets`, CONSTRAINT `pets_ibfk_1` FOREIGN KEY (`ownerId`) REFERENCES `people` (`id`))
```

こののは、を`people`からし、 `InnoDB`の`ON DELETE`をしてをすることです。

```
DELETE FROM people
WHERE name = 'Paul';
```

2がされました
PaulがPeopleからされました
スポットはペットのカスケードでされます

のは、foreingキーのチェックをににすることです。

```
SET foreign_key_checks = 0;
DELETE p1, p2 FROM people p1 JOIN pets p2 ON p2.ownerId = p1.id WHERE p1.name = 'Paul';
SET foreign_key_checks = 1;
```

な

```
DELETE FROM `myTable` WHERE `someColumn` = 'something'
```

`WHERE`はオプションですが、それがなければすべてのがされます。

DELETEとTRUNCATE

```
TRUNCATE tableName;
```

これにより、すべてのデータがされ、 `AUTO_INCREMENT`インデックスがリセットされます。なデータセットの`DELETE FROM tableName`から`DELETE FROM tableName`よりもはるかにです。これは、/テストににちます。

テーブルをりてると 、SQL Serverはデータをしませんが、テーブルをしてするため、ページのりてがされ、きされたページのにりてられたデータをできます。 このスペースは、`innodb_file_per_table=OFF`ためにただちにりすことはできません。

マルチテーブルDELETE

MySQLは、するをどのテーブルからするがあるかをできます

```
    -- remove only the employees
    DELETE e
    FROM Employees e JOIN Department d ON e.department_id = d.department_id
    WHERE d.name = 'Sales'
```

```
    -- remove employees and department
    DELETE e, d
    FROM Employees e JOIN Department d ON e.department_id = d.department_id
    WHERE d.name = 'Sales'
```

```
    -- remove from all tables (in this case same as previous)
    DELETE
```

```
    FROM Employees e JOIN Department d ON e.department_id = d.department_id
    WHERE d.name = 'Sales'
```

オンラインでをむ https://riptutorial.com/ja/mysql/topic/1487/

59: の

Examples

の

をするはのとおりです。

1. SETをして、をの、、にできます

   EXSET @var_string = 'my_var'; SET @var_num = '2' SET @var_date = '2015-07-20';

2. のようにselectのになるようにをすることができます

   EX@var= '123'をします。 SETをしないをりてるは、=をするがあります。これは、のステートメントselect、update ...では "="をしてするため、 = "、これは"これはではない、これはSETです "とっている

3. INTOをして、SELECTステートメントのになるようにをすることができます

   これは、クエリをうパーティションをにするがあるときににちました

EXSET @start_date = '2015-07-20'; SET @end_date = '2016-01-31';

```
#this gets the year month value to use as the partition names
SET @start_yearmonth = (SELECT EXTRACT(YEAR_MONTH FROM @start_date));
SET @end_yearmonth = (SELECT EXTRACT(YEAR_MONTH FROM @end_date));

#put the partitions into a variable
SELECT GROUP_CONCAT(partition_name)
FROM information_schema.partitions p
WHERE table_name = 'partitioned_table'
AND SUBSTRING_INDEX(partition_name,'P',-1) BETWEEN @start_yearmonth AND @end_yearmonth
INTO @partitions;

#put the query in a variable. You need to do this, because mysql did not recognize my variable
as a variable in that position. You need to concat the value of the variable together with the
rest of the query and then execute it as a stmt.
SET @query =
CONCAT('CREATE TABLE part_of_partitioned_table (PRIMARY KEY(id))
SELECT partitioned_table.*
FROM partitioned_table PARTITION(', @partitions,')
JOIN users u USING(user_id)
WHERE date(partitioned_table.date) BETWEEN ', @start_date,' AND ', @end_date);

#prepare the statement from @query
PREPARE stmt FROM @query;
#drop table
DROP TABLE IF EXISTS tech.part_of_partitioned_table;
#create table using statement
EXECUTE stmt;
```

Selectでをしてとグループをする

のようなteam_personテーブルがあるとします。

```
+======+===========+
| team |    person |
+======+===========+
|   A  |      John |
+------+-----------+
|   B  |     Smith |
+------+-----------+
|   A  |    Walter |
+------+-----------+
|   A  |     Louis |
+------+-----------+
|   C  | Elizabeth |
+------+-----------+
|   B  |     Wayne |
+------+-----------+


CREATE TABLE team_person AS SELECT 'A' team, 'John' person
UNION ALL  SELECT 'B' team,  'Smith' person
UNION ALL  SELECT 'A' team,  'Walter' person
UNION ALL  SELECT 'A' team,  'Louis' person
UNION ALL  SELECT 'C' team,  'Elizabeth' person
UNION ALL  SELECT 'B' team,  'Wayne' person;
```

テーブルをするにはteam_personしてrow_numberいずれかで、コラム

```
SELECT @row_no := @row_no+1 AS row_number, team, person
FROM team_person, (SELECT @row_no := 0) t;
```

または

```
SET @row_no := 0;
SELECT  @row_no := @row_no + 1 AS row_number, team, person
FROM team_person;
```

のをします

```
+============+======+===========+
| row_number | team |    person |
+============+======+===========+
|          1 |   A  |      John |
+------------+------+-----------+
|          2 |   B  |     Smith |
+------------+------+-----------+
|          3 |   A  |    Walter |
+------------+------+-----------+
|          4 |   A  |     Louis |
+------------+------+-----------+
|          5 |   C  | Elizabeth |
+------------+------+-----------+
|          6 |   B  |     Wayne |
```

```
+------------+------+-----------+
```

に、`team`によって`row_number`グループをするは

```
SELECT @row_no := IF(@prev_val = t.team, @row_no + 1, 1) AS row_number
   ,@prev_val := t.team AS team
   ,t.person
FROM team_person t,
  (SELECT @row_no := 0) x,
  (SELECT @prev_val := '') y
ORDER BY t.team ASC,t.person DESC;

+============+======+===========+
| row_number | team |    person |
+============+======+===========+
|          1 |   A  |    Walter |
+------------+------+-----------+
|          2 |   A  |     Louis |
+------------+------+-----------+
|          3 |   A  |      John |
+------------+------+-----------+
|          1 |   B  |     Wayne |
+------------+------+-----------+
|          2 |   B  |     Smith |
+------------+------+-----------+
|          1 |   C  | Elizabeth |
+------------+------+-----------+
```

オンラインでのをむ https://riptutorial.com/ja/mysql/topic/5013/の

60: マッピングテーブル

- このテーブルのAUTO_INCREMENT IDの - えられたPKは 'な' PKです。にはなはありません。
- MEDIUMINT - これは、すべてのINTsをなほどさいものにするがあることをいさせるものですさな⇒よりく。もちろんここのは、リンクののとしていなければなりません。
- UNSIGNED - ほぼすべてのINTがとされるもあります
- NOT NULL - まあ、そうですね。
- InnoDB - InnoDBのデータでPRIMARY KEYがクラスタリングされるため、MyISAMよりもです。
- INDEX(y_id, x_id) - PRIMARY KEYをINDEX(y_id, x_id)とににむことができます。のをにする。UNIQUEなはありません。それはINSERTsなをINSERTsます。
- セカンダリインデックスでは、にx_idするため、 INDEX(y_id)というだけでx_idます。しかし、はむしろ、が「カバーする」をんでいることをよりにしたいとう。

テーブルにをすることができます。これはまれです。なは、がすにするをできます。

FOREIGN KEYをすることができます。

Examples

なスキーマ

```
CREATE TABLE XtoY (
    # No surrogate id for this table
    x_id MEDIUMINT UNSIGNED NOT NULL,   -- For JOINing to one table
    y_id MEDIUMINT UNSIGNED NOT NULL,   -- For JOINing to the other table
    # Include other fields specific to the 'relation'
    PRIMARY KEY(x_id, y_id),            -- When starting with X
    INDEX      (y_id, x_id)             -- When starting with Y
) ENGINE=InnoDB;
```

については、のを。

オンラインでマッピングテーブルをむ https://riptutorial.com/ja/mysql/topic/4857/マッピングテーブル

61:

- DISTINCTとGROUP BYをじSELECTでしないでください。
- OFFSETをしてペ―ジしないでください。
- WHEREa、b=22,33はされません。
- UNIONのにALLまたはDISTINCTをにう - よりいALLまたはよりいDISTINCTをすることをいさせます。
- SELECT *をしないでください。に、なTEXTまたはBLOBがあるは、SELECT *をしないでください。 tmpテ―ブルとにオ―バ―ヘッドがあります。
- GROUP BYとORDER BYがまったくじリストをつことができるは、よりです。
- FORCE INDEXをしないでください。はけになるかもしれないが、はおそらくつくだろう。

ORDER BY、LIKE、REGEXPなどのディスカッションもしてください。リンクやそののトピックのがです。

なインデックスをするためのクックブック 。

Examples

しいインデックスをする

これはきなですが、もな「パフォ―マンス」のでもあります。

のためのなは、 ""をふことです。ここにながあります

```
INDEX(last_name, first_name)
```

これらのためにれています

```
WHERE last_name = '...'
WHERE first_name = '...' AND last_name = '...'   -- (order in WHERE does not matter)
```

しかし、そうではない

```
WHERE first_name = '...'   -- order in INDEX _does_ matter
WHERE last_name = '...' OR first_name = '...'   -- "OR" is a killer
```

キャッシュをしくする

```
innodb_buffer_pool_size
```

は、なRAMの70にするがあります。

なをける

```
x IN ( SELECT ... )
```

JOIN

であれば、 ORけてください。

WHERE DATE(x) = ...ようなのインデックスきのを "にしないでください。 WHERE x = ...とWHERE x = ...

、なをうことで、 WHERE LCASE(name1) = LCASE(name2)けることができます。

「ペ―ジり」にはOFFSETをしないでください。わりに「したをえてください」。

SELECT * ...けてくださいSELECT * ... デバッグしないり。

Maria Deleva、Barranka、Batsuへのこれはプレ―スホルダ―です。なをするには、これらのをしてください。あなたができることをやった、はりのをし、そして/またはそれらをげるためにします。

ネガティブ

パフォ―マンスにたないのあるものがいくつかあります。くなったやにしています。

- InnoDBは、MyISAMがよりくなるとはえないほどされました。
- PARTITIONingはパフォ―マンスのメリットはほとんどありません。パフォ―マンスをなうもあります。
- query_cache_size 100Mよりきくすると、はパフォ―マンスがします。
- my.cnfのをやすと、「スワッピング」がするがあります。これはなパフォ―マンスのです。
- 「」 INDEX(foo(20)) はににたない。
- OPTIMIZE TABLEはほとんどににたない。 そしてそれはテ―ブルをロックするがあります。

インデックスがある

さなテ―ブルでクエリをするためにもなことは、なインデックスをつことです。

```
WHERE a = 12  --> INDEX(a)
WHERE a > 12  --> INDEX(a)

WHERE a = 12 AND b > 78  --> INDEX(a,b) is more useful than INDEX(b,a)
WHERE a > 12 AND b > 78  --> INDEX(a) or INDEX(b); no way to handle both ranges

ORDER BY x  --> INDEX(x)
ORDER BY x, y  --> INDEX(x,y) in that order
ORDER BY x DESC, y ASC  --> No index helps - because of mixing ASC and DESC
```

にさないでください

よくあるいは、びしのでインデックスきのをすことです。たとえば、これはによってけられません。

```
WHERE DATE(dt) = '2000-01-01'
```

わりに、 `INDEX(dt)`すると、インデックスをできます。

```
WHERE dt = '2000-01-01'  -- if `dt` is datatype `DATE`
```

これは`DATE` 、 `DATETIME` 、 `TIMESTAMP` 、さらには`DATETIME(6)` マイクロでします。

```
WHERE dt >= '2000-01-01'
  AND dt  < '2000-01-01' + INTERVAL 1 DAY
```

または

に`OR`はをします。

```
WHERE a = 12 OR b = 78
```

`INDEX(a,b)`することはできず`INDEX(a,b)` "マ―ジ"をして`INDEX(a)`, `INDEX(b)`をするとしないがあります。インデックスのマ―ジはもありませんが、ほんのかです。

```
WHERE x = 3 OR x = 5
```

にする

```
WHERE x IN (3, 5)
```

そのに`x`をつをすることがあります。

サブクエリ

サブクエリにはいくつかのがあり、のがなります。まず、サブクエリは「」または「なし」のいずれかになることにしてください。は、サブクエリのからのにすることをします。これは、に、ごとにサブクエリをするがあることをします。

このサブクエリはしばしばかなりいです。1つのをすがあります。これは、 `LEFT JOIN`よりもずしもではないが、としてであることがい。

```
SELECT a, b, ( SELECT ... FROM t WHERE t.x = u.x ) AS c
    FROM u ...
SELECT a, b, ( SELECT MAX(x) ... ) AS c
    FROM u ...
```

```
SELECT a, b, ( SELECT x FROM t ORDER BY ... LIMIT 1 ) AS c
    FROM u ...
```

これはです

```
SELECT ...
    FROM ( SELECT ... ) AS a
    JOIN b ON ...
```

FROM-SELECTにする

- それが1をす、らしい。
- (SELECT @n := 0)が(SELECT @n := 0)であるため、りのまたはクエリでするための@をするといういパラダイムやはり "1"があります。
- くのをし、 JOINもくのをつ(SELECT ...)、はひどくなるがあります。 5.6よりはインデックスがなかったので、 CROSS JOINになりました。 5.6+には、テンポラリテ―ブルののインデックスをし、それをします。これは、 SELECTしたときにだけスロ―します。

JOIN + GROUP BY

なクエリにつながるなは、のようなものになります。

```
SELECT ...
    FROM a
    JOIN b  ON ...
    WHERE ...
    GROUP BY a.id
```

まず、 JOINはをします。 GROUP BYそれをaのにします。

この - のをするいはありません。 1つのなオプションは、 JOINをSELECTのサブクエリにすることです。これにより、 GROUP BYもされます。

オンラインでをむ https://riptutorial.com/ja/mysql/topic/4292/

62:

パラメーター

ASCII	ののをします。
BIN	のバイナリをむをします。
BIT_LENGTH	のさをビットです
CHAR	されたのをす
CHAR_LENGTH	のをします。
CHARACTER_LENGTH	CHAR_LENGTHの
CONCAT	されたをす
CONCAT_WS	セパレーータとのをす
ELT	インデックスのをす
EXPORT_SET	のビットにされたすべてのビットにして、onをし、すべてのされていないビットにして、offをするようなをします
フィールド	ののののインデックスをします。
FIND_IN_SET	2のののののインデックスをします。
フォーマット	されたののをします。
FROM_BASE64	ベース64にデコードしてをす
HEX	またはのの16をします。
インサート	されたに、されたまでをする
INSTR	ののインデックスをします。
LCASE	LOWERの
	されたののをします。
さ	のさをバイトでします。
き	なパターンマッチング

LOAD_FILE	されたファイルをロードする
LOCATE	ののをします。
LOWER	をでします。
LPAD	されたでパディングされたをします。
LTRIM	のスペースをする
MAKE_SET	コンマでられたをす
	をする
MID	されたからまるをします。
みたいではなく	なパターンマッチングの
NOT REGEXP	REGEXPの
OCT	の8をむをします。
OCTET_LENGTH	LENGTHの
ORD	ののコードをします。
ポジション	LOCATEの
もり	SQLでするをエスケープする
REGEXP	をしたパターンマッチング
りす	しただけをりします。
REPLACE	したの
	のをさせる
	されたのをします。
RLIKE	REGEXPの
RPAD	しただけをする
RTRIM	のスペースをする
SOUNDEX	soundexをす
のようにこえる	サウンドをする

スペース	されたスペースのをします。
STRCMP	2つのをする
SUBSTR	されたをします。
SUBSTRING	されたをします。
SUBSTRING_INDEX	りがされたにからをす
TO_BASE64	をbase-64にしてします。
トリム	とのスペースをする
UCASE	UPPERの
UNHEX	の16をむをします。
アッパー	に
WEIGHT_STRING	のみをします。

Examples

カンマりリストでをする

```
SELECT FIND_IN_SET('b','a,b,c');
```

り

    2

```
SELECT FIND_IN_SET('d','a,b,c');
```

り

    0

STR_TO_DATE - をにする

[] `07/25/2016`ようなをつ`my_date_field`というののをすると、のは`STR_TO_DATE`のを`STR_TO_DATE`ます。

```
SELECT STR_TO_DATE(my_date_field, '%m/%d/%Y') FROM my_table;
```

このを`WHERE`のとしてすることもできます。

LOWER/ LCASE

をにする

LOWER

```
LOWER('fOoBar') -- 'foobar'
LCASE('fOoBar') -- 'foobar'
```

REPLACE

をにする

REPLACEstr、from_str、to_str

```
REPLACE('foobarbaz', 'bar', 'BAR') -- 'fooBARbaz'
REPLACE('foobarbaz', 'zzz', 'ZZZ') -- 'foobarbaz'
```

SUBSTRING

SUBSTRINGまたはのものSUBSTRは、されたからし、オプションでされたさのをします

```
SUBSTRING(str, start_position)
```

```
SELECT SUBSTRING('foobarbaz', 4); -- 'barbaz'
SELECT SUBSTRING('foobarbaz' FROM 4); -- 'barbaz'

-- using negative indexing
SELECT SUBSTRING('foobarbaz', -6); -- 'barbaz'
SELECT SUBSTRING('foobarbaz' FROM -6); -- 'barbaz'
```

```
SUBSTRING(str, start_position, length)
```

```
SELECT SUBSTRING('foobarbaz', 4, 3); -- 'bar'
SELECT SUBSTRING('foobarbaz', FROM 4 FOR 3); -- 'bar'

-- using negative indexing
SELECT SUBSTRING('foobarbaz', -6, 3); -- 'bar'
SELECT SUBSTRING('foobarbaz' FROM -6 FOR 3); -- 'bar'
```

UPPER/ UCASE

をにする

UPPER

```
UPPER('fOoBar') -- 'FOOBAR'
UCASE('fOoBar') -- 'FOOBAR'
```

さ

のさをバイトでします。いくつかのはのバイトをしてエンコードされるかもしれないので、さをでするにはCHAR_LENGTHをしてください。

LENGTH

```
LENGTH('foobar') -- 6
LENGTH('fööbar') -- 8 -- contrast with CHAR_LENGTH(...) = 6
```

CHAR_LENGTH

のをします。

CHAR_LENGTH

```
CHAR_LENGTH('foobar') -- 6
CHAR_LENGTH('fööbar') -- 6 -- contrast with LENGTH(...) = 8
```

HEX

を16にします。これはにされます。

```
HEX('fööbar') -- 66F6F6626172 -- in "CHARACTER SET latin1" because "F6" is hex for ö
HEX('fööbar') -- 66C3B6C3B6626172  -- in "CHARACTER SET utf8 or utf8mb4" because "C3B6" is hex
for ö
```

オンラインでをむ https://riptutorial.com/ja/mysql/topic/1399/

63: しいユーザーをする

MySQLユーザーのリストをするには、のコマンドをします。

```
SELECT User,Host FROM mysql.user;
```

Examples

MySQLユーザをする

しいユーザーをするには、のなをするがあります。

ステップ1 `MySQL`に`root`としてログインする

```
$ mysql -u root -p
```

ステップ2 mysqlコマンドプロンプトがされます

```
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY 'test_password';
```

ここでは、しいユーザーをしましたが、このユーザーには`permissions`がありません。したがって、のコマンドをしてユーザーに`permissions`をりてます。

```
mysql> GRANT ALL PRIVILEGES ON my_db.* TO 'my_new_user'@'localhost' identified by
'my_password';
```

パスワードをする

なはのとおりです。

```
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY 'test_password';
```

ただし、パスワードをクリアテキストでハードコードすることをおめしないは、 `PASSWORD()`がすハッシュを`PASSWORD`ディレクティブをしてすることもできます。

```
mysql> select PASSWORD('test_password'); -- returns *4414E26EDED6D661B5386813EBBA95065DBC4728
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY PASSWORD
'*4414E26EDED6D661B5386813EBBA95065DBC4728';
```

ユーザーをし、すべてのをスキーマにする

```
grant all privileges on schema_name.* to 'new_user_name'@'%' identified by 'newpassword';
```

これはしいrootユーザーをするためにできます

ユーザーの

```
rename user 'user'@'%' to 'new_name`@'%';
```

ってユーザーをすると、そのユーザーのをできます

オンラインでしいユーザーをするをむ https://riptutorial.com/ja/mysql/topic/3508/しいユーザーをする

64: との

Examples

```
Select Now();
```

のサーバーのとをします。

```
Update `footable` set mydatefield = Now();
```

これにより、フィールド`mydatefield`が、サーバーのされたタイムゾーンでのサーバーのとにされます。

```
'2016-07-21 12:00:00'
```

```
NOW() + INTERVAL 1 DAY  -- This time tomorrow

CURDATE() - INTERVAL 4 DAY -- Midnight 4 mornings ago
```

310にされたmysqlをする180600

```
SELECT qId,askDate,minuteDiff
FROM
(   SELECT qId,askDate,
    TIMESTAMPDIFF(MINUTE,askDate,now()) as minuteDiff
    FROM questions_mysql
) xDerived
WHERE minuteDiff BETWEEN 180 AND 600
ORDER BY qId DESC
LIMIT 50;

+----------+---------------------+------------+
| qId      | askDate             | minuteDiff |
+----------+---------------------+------------+
| 38546828 | 2016-07-23 22:06:50 |        182 |
| 38546733 | 2016-07-23 21:53:26 |        195 |
| 38546707 | 2016-07-23 21:48:46 |        200 |
| 38546687 | 2016-07-23 21:45:26 |        203 |
| ...      |                     |            |
+----------+---------------------+------------+
```

`TIMESTAMPDIFF()` MySQLマニュアルページ。

MySQLのには`CURDATE() + 1`ようなをしないでください。に、Oracleデータベースにれているは、りのをしません。わりに`CURDATE() + INTERVAL 1 DAY`してください。

にするテスト

`BETWEEN`

... AND ...をにすることはにですが、があります。わりに、このパターンはほとんどのをします。

```
WHERE x >= '2016-02-25'
  AND x  < '2016-02-25' + INTERVAL 5 DAY
```

- BETWEENは「」BETWEEN、またはをみます。
- 23:59:59は、 DATETIMEマイクロのをっていれば、でっています。
- このパターンは、やそののデータをけるためのものです。
- xがDATE、 DATETIME、 TIMESTAMPいずれであってもしTIMESTAMP。

SYSDATE、NOW、CURDATE

```
  SELECT SYSDATE();
```

このは、またはのどちらのコンテキストでされるかにじて、のとを'YYYY-MM-DD HH:MM:SS'またはYYYYMMDDHHMMSSでとしてします。のタイムゾーンでとをします。

```
  SELECT NOW();
```

このはSYSDATE()です。

```
  SELECT CURDATE();
```

このは、またはのどちらのコンテキストでされるかにじて、のをなしで、 'YYYY-MM-DD'またはYYYYMMDDのとしてします。のタイムゾーンのをします。

されたまたはのからをする

```
SELECT DATE('2003-12-31 01:02:03');
```

はのようになります。

```
2003-12-31
```

とののためのの

くののデータベースには、 DATETIMEまたはTIMESTAMPのがまたはをむくのにわたるくのがありTIMESTAMP。 WHEREをしてそのタイムパンのをするがあることがよくあります。たとえば、テーブルから201691ののをすることができます。

これをうなはこれです

```
 WHERE DATE(x) = '2016-09-01'   /* slow! */
```

これは、のにDATE()をするため、です。つまり、MySQLはxをべなければならず、インデックスは

できません。

をうためのよりいは、これです

```
WHERE x >= '2016-09-01'
  AND x <  '2016-09-01' + INTERVAL 1 DAY
```

これは、ののをしxまで、のにどこかにたわるが、それゆえめない <を。

テーブルにxカラムのインデックスがある、データベースサーバはインデックスにしてレンジスキャンをできます。つまり、xののするをすばやくつけて、にするがつかるまでにインデックスをスキャンすることができます。インデックスのスキャンは、 `DATE(x) = '2016-09-01`なスキャンよりもはるかに`DATE(x) = '2016-09-01` 。

よりにえても、これをするようにされないでください。

```
 WHERE x BETWEEN '2016-09-01' AND '2016-09-01' + INTERVAL 1 DAY  /*  wrong! */
```

これはレンジスキャンとじがありますが、201692のにxをつをします。これはあなたがむものではありません。

オンラインでとのをむ https://riptutorial.com/ja/mysql/topic/1882/との

65:

- UPDATE [LOW_PRIORITY] [IGNORE]テーブルSET column1 = expression1、column2 = expression2、... [WHERE]; //なのテーブルの
- UPDATE [LOW_PRIORITY] [IGNORE]テーブルSET column1 = expression1、column2 = expression2、... [WHERE] [ORDER BY[ASC | DESC]] [LIMIT_カウント]; // order byとlimitでする
- UPDATE [LOW_PRIORITY] [IGNORE]テーブル1、テーブル2、... SET column1 = expression1、column2 = expression2、... [WHERE]; //のテーブルの

Examples

な

1の

```
UPDATE customers SET email='luke_smith@email.com' WHERE id=1
```

このクエリは、コンテンツ`email`での`customers`にテーブル`luke_smith@email.com`の`id` 1.データベーステーブルのいものとしいがそれぞれに、にされているにしいです。

customers

id	firstname	lastname	email
1	Luke	Smith	luke@example.com
2	Anna	Carey	anna@example.com
3	Todd	Winters	todd@example.com

customers

id	firstname	lastname	email
1	Luke	Smith	luke_smith@email.com
2	Anna	Carey	anna@example.com
3	Todd	Winters	todd@example.com

すべてのの

```
UPDATE customers SET lastname='smith'
```

このクエリは、 `customers`テーブルのすべてのエントリの`lastname`のをします。データベーステーブルのいとしいは、それぞれのとにされています。

customers

id	firstname	lastname	email
1	Luke	Smith	luke@example.com
2	Anna	Carey	anna@example.com
3	Todd	Winters	todd@example.com

customers

id	firstname	lastname	email
1	Luke	Smith	luke@example.com
2	Anna	Smith	anna@example.com
3	Todd	Smith	todd@example.com

UPDATEクエリにはWHEREをするがあります。をしない、そののすべてのレコードがされます。のでは、customersテーブルのlastnameのしいSmithがすべてのにされています。

パターンによる

`LOAD DATA INFILE`からインポートされたCSVデータののバッチをす、 `questions_mysql`というと`iwtQuestions`インポートされたをえてみましょう。インポートにワークテーブルがりてられ、データがインポートされ、そのプロセスはここにはされません。

インポートされたワークテーブルデータへのをして、データをします。

```
UPDATE questions_mysql q -- our real table for production
join iwtQuestions i -- imported worktable
ON i.qId = q.qId
SET q.closeVotes = i.closeVotes,
q.votes = i.votes,
q.answers = i.answers,
q.views = i.views;
```

エイリアス`q`と`i`は、テーブルをするためにされます。これにより、とみやすさがになります。

`qId`は、キーで、StackoverflowのIDをします。 4つのは、からするのためにされます。

ORDER BYとLIMITをしたUPDATE

SQLで`ORDER BY`をすると、されたでがされます。

SQLで`LIMIT`をすると、なにがされます。 `LIMIT`がされていない、はありません。

`ORDER BY`と`LIMIT`は、マルチテーブルのにはできません。

`ORDER BY`と`LIMIT`をしたMySQL `UPDATE`は、

```
UPDATE [ LOW_PRIORITY ] [ IGNORE ]
tableName
SET column1 = expression1,
    column2 = expression2,
    ...
[WHERE conditions]
[ORDER BY expression [ ASC | DESC ]]
[LIMIT row_count];

---> Example
UPDATE employees SET isConfirmed=1 ORDER BY joiningDate LIMIT 10
```

のでは、 `joiningDate`するのにって10がされます。

のの

のの`UPDATE`では、をたすされたのをします。するは、がするでも、されます。

のテーブル`UPDATE`では、 `ORDER BY`と`LIMIT`はできません。

マルチテーブル`UPDATE`は、

```
UPDATE [LOW_PRIORITY] [IGNORE]
table1, table2, ...
    SET column1 = expression1,
        column2 = expression2,
        ...
    [WHERE conditions]
```

たとえば、2つのテーブル、 `products` 、および`salesOrders`えてみ`salesOrders`。 そのは、ののを、すでにされているからします。その、々はまた、のにそのをやすがある`products`テーブル。これは、のようなのSQLでうことができます。

```
UPDATE products, salesOrders
  SET salesOrders.Quantity = salesOrders.Quantity - 5,
      products.availableStock = products.availableStock + 5
WHERE products.productId = salesOrders.productId
  AND salesOrders.orderId = 100 AND salesOrders.productId = 20;
```

のでは、 '5'は`salesOrders`テーブルから`salesOrders`され、 `WHERE`にって`products`テーブルでじがし`products`。

なるでのをするは、をするがはるかにです。

```
UPDATE people
SET name =
  (CASE id WHEN 1 THEN 'Karl'
           WHEN 2 THEN 'Tom'
           WHEN 3 THEN 'Mary'
   END)
WHERE id IN (1,2,3);
```

では、するごとに1つのクエリではなく、1つのクエリのみをサーバーにできます。ケースには`WHERE`でされるすべてのパラメータがまれているがあります。

オンラインでをむ https://riptutorial.com/ja/mysql/topic/2738/

66: とチューニング

は3つののいずれかでわれます。

- コマンドラインオプション
- `my.cnf`ファイル
- サーバーからをする

コマンドラインオプションのは、 `mysqld --long-parameter-name=value --another-parameter`です。 じパラメータを`my.conf`ファイルにれることができます。 のパラメータは、MySQLのシステムをしてできます。パラメータのなリストについては、をチェックしてください。

は、ダッシュをつことができます`-`またはアンダースコア`_`。 `=`りにスペースがするがあります。 `K`、 `M`、 `G`、キロ、メガ、ギガののきいをけることができます。 1に1つの。

Flags、 `ON`と`1`はですが、 `OFF`と`0`はじです。いくつかのフラグはにもない。

`my.cnf`にをくとき、 サーバのすべてのは`[mysqld]`セクションになければならないので、をにファイルのにしないでください。 のmysqlインスタンスが1つのmy.cnfをできるツールの、セクションはなるがあります。

Examples

InnoDBのパフォーマンス

my.cnfにできるはもあります。 MySQLの 'lite'ユーザにとって、それほどではありません。

データベースがになったら、のパラメータをすることをおめします。

```
innodb_buffer_pool_size
```

これは、 な RAMの70にするがありますRAMが4GBの、さなVMまたはアンティークマシンをしているは、よりさなパーセンテージ。このは、InnoDB ENGINEでされるキャッシュのをします。したがって、InnoDBのパフォーマンスにとってはにです。

なデータをできるパラメータ

にイメージやビデオをするがあるは、アプリケーションによってにじてをするがあります

max_allowed_packet = 10M

MはMb、GはGb、KはKb

group_concatのをやす

group_concatは、 groupのnullのをするためにされます。 ののは、 group_concat_max_lenオプションをしてできます。

```
SET [GLOBAL | SESSION] group_concat_max_len = val;
```

GLOBALをするとながわれますが、 SESSIONをするとのセッションのがされます。

のInnoDB

これは、InnoDBテーブルをするMySQLサーバーののです。 InnoDBをすると、クエリキャッシュはありません。またはデータベースがDROPときにディスク・スペースをします。 SSDをしている、フラッシュはですSDDはではありません。

```
default_storage_engine = InnoDB
query_cache_type = 0
innodb_file_per_table = 1
innodb_flush_neighbors = 0
```

innodb_thread_concurrencyを0にinnodb_thread_concurrencyことで、デフォルトの4スレッドをできることをしてください。これによりInnoDBはなにづいてすることができます。

```
innodb_thread_concurrency = 0
innodb_read_io_threads = 64
innodb_write_io_threads = 64
```

ハードドライブの

MySQLのIOPSののとcapacity_maxをします。 HDDのデフォルトは200ですが、ではのIOPSがなSSDではこのをしたいとえています。 IOPSをするためにできるテストはたくさんあります。 の *MySQL* サーバーをしているは 、のはほぼになるはずです。じマシンでのサービスをしているは 、にじてするがあります。

```
innodb_io_capacity = 2500
innodb_io_capacity_max = 3000
```

RAM

MySQLになRAMをします。は7080ですが、これはにインスタンスがMySQLであるかどうか、なRAMのによってなります。たくさんあるはRAMつまりリソースをにしないでください。

```
innodb_buffer_pool_size = 10G
```

なMySQL

デフォルトのaes-128-ecbでは、ECBElectronic Codebookモードがされています。これはではなく、してしないでください。わりに、ファイルにのをします。

```
block_encryption_mode = aes-256-cbc
```

オンラインでとチューニングをむ https://riptutorial.com/ja/mysql/topic/3134/とチューニング

67:

き

は、なのパターンをするなです。

Examples

REGEXP / RLIKE

`REGEXP` またはそのシノニム、 `RLIKE` は、にづいたパターンマッチングをにします。

の`employee`テーブルをえてみましょう。

```
+-------------+-------------+-------------+--------------+----------+
| EMPLOYEE_ID | FIRST_NAME  | LAST_NAME   | PHONE_NUMBER | SALARY   |
+-------------+-------------+-------------+--------------+----------+
|         100 | Steven      | King        | 515.123.4567 | 24000.00 |
|         101 | Neena       | Kochhar     | 515.123.4568 | 17000.00 |
|         102 | Lex         | De Haan     | 515.123.4569 | 17000.00 |
|         103 | Alexander   | Hunold      | 590.423.4567 |  9000.00 |
|         104 | Bruce       | Ernst       | 590.423.4568 |  6000.00 |
|         105 | David       | Austin      | 590.423.4569 |  4800.00 |
|         106 | Valli       | Pataballa   | 590.423.4560 |  4800.00 |
|         107 | Diana       | Lorentz     | 590.423.5567 |  4200.00 |
|         108 | Nancy       | Greenberg   | 515.124.4569 | 12000.00 |
|         109 | Daniel      | Faviet      | 515.124.4169 |  9000.00 |
|         110 | John        | Chen        | 515.124.4269 |  8200.00 |
+-------------+-------------+-------------+--------------+----------+
```

パターン^

`FIRST_NAME`が**N**でまるすべてのをします。

クエリ

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^N'
-- Pattern start with----------------------------^
```

パターン$ **

`PHONE_NUMBER`が**4569**でわるすべてのをします。

クエリ

```
SELECT * FROM employees WHERE PHONE_NUMBER REGEXP '4569$'
-- Pattern end with-----------------------------------^
```

NOT REGEXP

`FIRST_NAME` が **N**で`FIRST_NAME` ないをすべてします。

クエリ

```
SELECT * FROM employees WHERE FIRST_NAME NOT REGEXP '^N'
-- Pattern does not start with---------------^
```

がまれています

`LAST_NAME`がまれ 、 `FIRST_NAME`はがまれて`a`すべてのをします。

クエリ

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP 'a' AND LAST_NAME REGEXP 'in'
-- No ^ or $, pattern can be anywhere ------------------------------------^
```

[]とののの

`FIRST_NAME`が**A**または**B**または**C**でまるすべてのをします。

クエリ

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^[ABC]'
-------------------------------------------------^^---^
```

パターンまたは|

`FIRST_NAME`が**A**または**B**または**C**でまり、 **r** 、 **e** 、または**i**でわるすべてのをします。

クエリ

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^[ABC]|[rei]$'
-- -------------------------------------------------^^---^^^---^^
```

のをえる

のクエリをえてみましょう。

```
SELECT FIRST_NAME, FIRST_NAME REGEXP '^N' as matching FROM employees
```

`FIRST_NAME REGEXP '^N'`は、 `FIRST_NAME ^N`するというにじて*1*または*0*です。

それをよりよくするには

```
SELECT
FIRST_NAME,
IF(FIRST_NAME REGEXP '^N', 'matches ^N', 'does not match ^N') as matching
FROM employees
```

に、するとしないのをのようにカウントします。

```
SELECT
IF(FIRST_NAME REGEXP '^N', 'matches ^N', 'does not match ^N') as matching,
COUNT(*)
FROM employees
GROUP BY matching
```

オンラインでをむ https://riptutorial.com/ja/mysql/topic/9444/

68: のをつ

のデータをつカラムをするには、MySQLバージョン5.6.4がです。

たとえば、 `DATETIME(3)` はタイムスタンプでミリのを、 `TIMESTAMP(6)` は* nixスタイルのタイムスタンプでマイクロのを`DATETIME(3)`ます。

これをんでください http : //dev.mysql.com/doc/refman/5.7/en/fractional-seconds.html

`NOW(3)` はMySQLサーバーのオペレーティングシステムからのをミリでします。

* 0.001のようなMySQLのはにIEEE754としてわれるので、Sunがいになるにをうことはありません。

Examples

ミリのでのをする

```
SELECT NOW(3)
```

トリックをう。

Javascriptのタイムスタンプのようなでのをします。

Javascriptタイムスタンプは、しいUNIXの`time_t`データについており、 `1970-01-01 00:00:00` UTCのミリをし`1970-01-01 00:00:00`。

このはのをJavascriptのタイムスタンプとしてします。 これは、のtime_zoneになくしくされます。

```
 ROUND(UNIX_TIMESTAMP(NOW(3)) * 1000.0, 0)
```

`TIMESTAMP`がにされているは、UNIX_TIMESTAMPをしてJavascriptタイムスタンプとしてできます。

```
  SELECT ROUND(UNIX_TIMESTAMP(column) * 1000.0, 0)
```

に`DATETIME`がまれていて、そのをJavascriptタイムスタンプとしてすると、それらのタイムスタンプはされているタイムゾーンのタイムゾーンオフセットによってオフセットされます。

サブのをするをつテーブルをします。

```
CREATE TABLE times (
     dt DATETIME(3),
```

```
    ts TIMESTAMP(3)
);
```

ミリの/フィールドをつテーブルをします。

```
INSERT INTO times VALUES (NOW(3), NOW(3));
```

ミリの`NOW()`をむをテーブルにします。

```
INSERT INTO times VALUES ('2015-01-01 16:34:00.123','2015-01-01 16:34:00.128');
```

のミリのをします。

このをしてのをするは、 `NOW()`ではなく`NOW(3)`するがあることにしてください。

ミリの/をテキストにします。

`%f`は、 DATE_FORMATのです。

```
SELECT DATE_FORMAT(NOW(3), '%Y-%m-%d %H:%i:%s.%f')
```

マイクロで`2016-11-19 09:52:53.248000`ようなをします。 `NOW(3)`をしたため、のの3は0です。

Javascriptタイムスタンプを TIMESTAMPにする

Javascriptのタイムスタンプ `1478960868932`などがあるは、そのをMySQLののにできます。

```
FROM_UNIXTIME(1478960868932 * 0.001)
```

こののをして、Javascriptタイムスタンプを MySQLテーブルにするのはです。これをう

```
INSERT INTO table (col) VALUES (FROM_UNIXTIME(1478960868932 * 0.001))
```

らかに、のをするがあります。

オンラインでのをつをむ https://riptutorial.com/ja/mysql/topic/7850/のをつ

69:

MySQLは、ほとんどのマシンで、に64ビットIEEE 754をします。

コンテキストでは、をします。

- `RAND()`はなジェネレーターではありません。にをくするためにされます

Examples

MySQLはのをします

オペレーター		
+		`SELECT 3+5;` →8 `SELECT 3.5+2.5;` - > 6.0 `SELECT 3.5+2;` →5.5
-		`SELECT 3-5;` →-2
*		`SELECT 3 * 5;` - > 15
/		`SELECT 20 / 4;` →5 `SELECT 355 / 113;` - > 3.1416 `SELECT 10.0 / 0;` - > NULL
DIV		`SELECT 5 DIV 2;` →2
%または`MOD`	モジュロ	`SELECT 7 % 3;` - > 1 `SELECT 15 MOD 4` - > 3 `SELECT 15 MOD -4` - > 3 `SELECT -15 MOD 4` - > -3 `SELECT -15 MOD -4` - > -3 `SELECT 3 MOD 2.5` - > 0.5

BIGINT

のがすべての、MySQLは`BIGINT`き64ビットデータをしてをいます。えば

`select (1024 * 1024 * 1024 * 1024 *1024 * 1024) + 1` 1,152,921,504,606,846,977をします

そして

`select (1024 * 1024 * 1024 * 1024 *1024 * 1024 * 1024` → `BIGINT`エラー

ダブル

のがの、MySQLは64ビットIEEE 754をします。くのはになではなくなので、をするときはがです 。

Pi

のは、 `PI`のを6にします。のは`DOUBLE`しています。

```
SELECT PI();    -> 3.141593
```

SIN、COS

は、ではなくラジアンです。すべてのは、 IEEE 754 64ビットでわれます。すべてのはεとばれるさなのをけますので、それらをでしようとしないでください。にこれらのエラーをするはありません。らはにみまれています。

`DECIMAL`をでると、ににされた、にります。

ラジアンでされるXのをします。

```
SELECT SIN(PI()); -> 1.2246063538224e-16
```

XがラジアンでえられたときのXのをす

```
SELECT COS(PI()); -> -1
```

ラジアンでされるXのをします。はゼロににいが、にゼロではないことにしてください。これはマシンεのです。

```
SELECT TAN(PI());   -> -1.2246063538224e-16
```

アークコサインコサイン

Xがであればxのアークコサインをし`-1 to 1`

```
SELECT ACOS(1);    -> 0
SELECT ACOS(1.01); -> NULL
```

アークサインサイン

Xが`-1 to 1`にある、Xのをします。

```
SELECT ASIN(0.2); -> 0.20135792079033
```

アークタンジェントタンジェント

ATAN(x)は、ののをします。

```
SELECT ATAN(2); -> 1.1071487177941
```

ATAN2(X, Y)は、2つのXとYのアークタンジェントをします.Y / Xのアークタンジェントをするのとていますが、にはですATAN2(X, Y)がゼロにく、ののをして、のをします。

なり、 ATAN()ではなくATAN2()をするをすることをおATAN2()します。

```
 ATAN2(1,1);    -> 0.7853981633974483 (45 degrees)
 ATAN2(1,-1);   -> 2.356194490192345  (135 degrees)
 ATAN2(0, -1);  -> PI  (180 degrees)  don't try ATAN(-1 / 0)... it won't work
```

コタンジェント

Xのコタンジェントをします。

```
SELECT COT(12); -> -1.5726734063977
```

```
SELECT RADIANS(90) -> 1.5707963267948966
SELECT SIN(RADIANS(90)) -> 1
SELECT DEGREES(1), DEGREES(PI()) -> 57.29577951308232, 180
```

めROUND、FLOOR、CEIL

10をにめます。

なえばDECIMAL のののが5の、このはを0かられたのにめます 。そののが4の、このはゼロにもいのにめます 。

```
SELECT ROUND(4.51) -> 5
SELECT ROUND(4.49) -> 4
SELECT ROUND(-4.51) -> -5
```

DOUBLE の ROUND()のはCライブラリーによってなります。くのシステムでは、これはROUND() がもいのルールへのラウンドをすることをします。

```
SELECT ROUND(45e-1) -> 4  -- The nearest even value is 4
SELECT ROUND(55e-1) -> 6  -- The nearest even value is 6
```

をりげる

を`CEIL()`は、 `CEIL()`または`CEILING()`をします

```
SELECT CEIL(1.23)    -> 2
SELECT CEILING(4.83) -> 5
```

をりてる

を`FLOOR()`には、 `FLOOR()`をします

```
SELECT FLOOR(1.99) -> 1
```

フロアとCEILは、 - にづく/れる

```
SELECT FLOOR(-1.01), CEIL(-1.01) -> -2 and -1
SELECT FLOOR(-1.99), CEIL(-1.99) -> -2 and -1
```

10をのにめます。

```
SELECT ROUND(1234.987, 2) -> 1234.99
SELECT ROUND(1234.987, -2) -> 1200
```

きときのと「5」もされます。

をパワ—POWにげる

xをyにするには、 `POW()` `POWER()`または`POWER()`をします

```
SELECT POW(2,2); => 4
SELECT POW(4,2); => 16
```

SQRT

`SQRT()`をします。 がのは`NULL`が`NULL`

```
SELECT SQRT(16); -> 4
SELECT SQRT(-3); -> NULL
```

RAND

をする

0と1のをするには、 RAND()をします

のクエリがあるとします

```
SELECT i, RAND() FROM t;
```

これはのようなものをします

	RAND
1	0.6191438870682
2	0.93845168309142
3	0.83482678498591

の

a <= n <= bののをするには、のをできます

```
FLOOR(a + RAND() * (b - a + 1))
```

たとえば、これは7と12ののをします

```
SELECT FLOOR(7 + (RAND() * 6));
```

テーブルのをランダムにすなはのとおりです。

```
SELECT * FROM tbl ORDER BY RAND();
```

これらはです。

MySQLのは、にではありません。つまり、MySQLをしてとしてするをすると、MySQLをしていることをっているなは、じるよりもにをすることができます。

とABS、SIGN

のをす

```
SELECT ABS(2);   -> 2
SELECT ABS(-46); -> 46
```

のsignは0とされます。

-1	n <0	SELECT SIGN(42); - > 1

0	n = 0	`SELECT SIGN(0);` →0
1	n> 0	`SELECT SIGN(-3);` - > -1

```
SELECT SIGN(-423421); -> -1
```

オンラインでをむ https://riptutorial.com/ja/mysql/topic/4516/

70: したルートパスワードからする

Examples

rootパスワードをし、ソケットユーザーとHTTPアクセスのrootユーザーをにする

をするパスワードをしてユーザーrootのアクセスがされました。YES mySQLをします。

```
sudo systemctl stop mysql
```

グラントテーブルをスキップしてmySQLをします。

```
sudo mysqld_safe --skip-grant-tables
```

ログイン

```
mysql -u root
```

SQLシェルでは、ユーザーがするかどうかをべます。

```
select User, password,plugin FROM mysql.user ;
```

ユーザーをしますすべてのプラグインでnullプラグインがです。

```
update mysql.user set password=PASSWORD('mypassword'), plugin = NULL WHERE User = 'root';
exit;
```

グラントテーブルをたないUnixシェルmySQLでは、グラントテーブルでしてください

```
sudo service mysql stop
sudo service mysql start
```

オンラインでしたルートパスワードからするをむ https://riptutorial.com/ja/mysql/topic/9973/したルートパスワードからする

71:

- `INNER`と`OUTER`はされます。
- `FULL`はMySQLではされていません。
- "commajoin" `FROM a,b WHERE ax=by` where `FROM a,b WHERE ax=by` はをひそめられています。わりに、 `FROM a JOIN b ON ax=by`してください。
- FROM a JOIN B ON ax = byのテーブルでするをむ。
- LEFT JOINからb ax = = byは、 `a`すべてのと`b`するデータをみ、するがないは`NULLs`みます。

Examples

DBのテーブルをするためのクエリ

```
CREATE TABLE `user` (
`id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(30) NOT NULL,
`course` smallint(5) unsigned DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB;

CREATE TABLE `course` (
`id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(50) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB;
```

InnoDBテーブルをしており、user.courseとcourse.idがしていることをっているので、キーをできます。

```
ALTER TABLE `user`
ADD CONSTRAINT `FK_course`
FOREIGN KEY (`course`) REFERENCES `course` (`id`)
ON UPDATE CASCADE;
```

クエリ

```
SELECT user.name, course.name
FROM `user`
INNER JOIN `course` on user.course = course.id;
```

サブクエリをつJOIN ""テーブル

```
SELECT x, ...
    FROM ( SELECT y, ... FROM ... ) AS a
    JOIN tbl  ON tbl.x = a.y
```

```
    WHERE ...
```

これは、サブクエリをテンポラリテーブルにし、にそれを`tbl JOIN`します。

5.6では、にをできませんでした。したがって、これはににでした。

```
SELECT ...
    FROM ( SELECT y, ... FROM ... ) AS a
    JOIN ( SELECT x, ... FROM ... ) AS b  ON b.x = a.y
    WHERE ...
```

5.6では、オプティマイザがなインデックスをつけ、にします。 これにはオーバーヘッドがあるため、まだではありません。

のなパラダイムは、かをするためのいわせをつことです

```
SELECT
        @n := @n + 1,
        ...
    FROM ( SELECT @n := 0 ) AS initialize
    JOIN the_real_table
    ORDER BY ...
```

これはには`ON`ないことによってされる`CROSS JOIN` デカルトですが、せは`the_real_table` nのにするがあるを1つだけすので`the_real_table`。

オーダーのあるの - テーマのバリエーション

これにより、すべてののすべてのがられます。

```
SELECT c.CustomerName, o.OrderID
    FROM Customers AS c
    INNER JOIN Orders AS o
        ON c.CustomerID = o.CustomerID
    ORDER BY c.CustomerName, o.OrderID;
```

これにより、のがカウントされます。

```
SELECT c.CustomerName, COUNT(*) AS 'Order Count'
    FROM Customers AS c
    INNER JOIN Orders AS o
        ON c.CustomerID = o.CustomerID
    GROUP BY c.CustomerID;
    ORDER BY c.CustomerName;
```

また、えますが、おそらくよりです

```
SELECT  c.CustomerName,
        ( SELECT COUNT(*) FROM Orders WHERE CustomerID = c.CustomerID ) AS 'Order Count'
    FROM Customers AS c
    ORDER BY c.CustomerName;
```

オーダーのあるのみをします。

```
SELECT  c.CustomerName,
    FROM Customers AS c
    WHERE EXISTS ( SELECT * FROM Orders WHERE CustomerID = c.CustomerID )
    ORDER BY c.CustomerName;
```

な

MySQLは`FULL OUTER JOIN`サポートしていませんが、エミュレートするはあります。

データの

```
-- ----------------------------
-- Table structure for `owners`
-- ----------------------------
DROP TABLE IF EXISTS `owners`;
CREATE TABLE `owners` (
`owner_id` int(11) NOT NULL AUTO_INCREMENT,
`owner` varchar(30) DEFAULT NULL,
PRIMARY KEY (`owner_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of owners
-- ----------------------------
INSERT INTO `owners` VALUES ('1', 'Ben');
INSERT INTO `owners` VALUES ('2', 'Jim');
INSERT INTO `owners` VALUES ('3', 'Harry');
INSERT INTO `owners` VALUES ('6', 'John');
INSERT INTO `owners` VALUES ('9', 'Ellie');
-- ----------------------------
-- Table structure for `tools`
-- ----------------------------
DROP TABLE IF EXISTS `tools`;
CREATE TABLE `tools` (
`tool_id` int(11) NOT NULL AUTO_INCREMENT,
`tool` varchar(30) DEFAULT NULL,
`owner_id` int(11) DEFAULT NULL,
PRIMARY KEY (`tool_id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1;
-- ----------------------------
-- Records of tools
-- ----------------------------
INSERT INTO `tools` VALUES ('1', 'Hammer', '9');
INSERT INTO `tools` VALUES ('2', 'Pliers', '1');
INSERT INTO `tools` VALUES ('3', 'Knife', '1');
INSERT INTO `tools` VALUES ('4', 'Chisel', '2');
INSERT INTO `tools` VALUES ('5', 'Hacksaw', '1');
INSERT INTO `tools` VALUES ('6', 'Level', null);
INSERT INTO `tools` VALUES ('7', 'Wrench', null);
INSERT INTO `tools` VALUES ('8', 'Tape Measure', '9');
INSERT INTO `tools` VALUES ('9', 'Screwdriver', null);
INSERT INTO `tools` VALUES ('10', 'Clamp', null);
```

をたいのですか

たちはがどのツ―ルをしているのか、そしてどのツ―ルがをたないのかをるためのリストをたいとえています。

クエリ

これをするために、 UNIONをして2つのクエリをみわせることができます。こののクエリでは、 LEFT JOINをしてのツ―ルにしています。これにより、すべてのがセットにされます。にツ―ルをしているかどうかはありません。

2のクエリでは、ツ―ルをにさせるためにRIGHT JOINをしています。このようにして、セットのすべてのツ―ルをできNULL。 がでない、にはにNULLがまれNULL。 することによりWHEREによってフィルタリングされ-clause owners.owner_id IS NULL々はののテ―ブルのデ―タをしているとして、すでに、のクエリによってされていないものをデ―タセットとしてをしています。

UNION ALLをしているので、2のクエリのセットはのクエリセットにアタッチされます。

```
SELECT `owners`.`owner`, tools.tool
FROM `owners`
LEFT JOIN `tools` ON `owners`.`owner_id` = `tools`.`owner_id`
UNION ALL
SELECT `owners`.`owner`, tools.tool
FROM `owners`
RIGHT JOIN `tools` ON `owners`.`owner_id` = `tools`.`owner_id`
WHERE `owners`.`owner_id` IS NULL;

+-------+--------------+
| owner | tool         |
+-------+--------------+
| Ben   | Pliers       |
| Ben   | Knife        |
| Ben   | Hacksaw      |
| Jim   | Chisel       |
| Harry | NULL         |
| John  | NULL         |
| Ellie | Hammer       |
| Ellie | Tape Measure |
| NULL  | Level        |
| NULL  | Wrench       |
| NULL  | Screwdriver  |
| NULL  | Clamp        |
+-------+--------------+
12 rows in set (0.00 sec)
```

3つのテ―ブルの

タグきのシンプルなウェブサイトにできる3つのテ―ブルがあるとしましょう。

- フィストテ―ブルはポストです。
- タグの2
- タグポストの3

のテ―ブル "ビデオゲ―ム"

id	タイトル	reg_date	コンテンツ
1	バイオショック·インフィニット	2016-08-08	....

"タグ"テーブル

id	
1	
2	エリザベス

"tags_meta"テーブル

post_id	tag_id
1	2

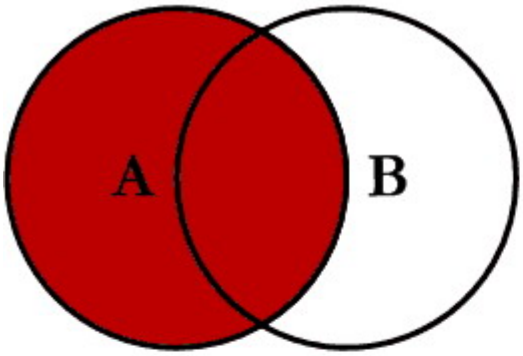
```
SELECT videogame.id,
    videogame.title,
    videogame.reg_date,
    tags.name,
    tags_meta.post_id
FROM tags_meta
INNER JOIN videogame ON videogame.id = tags_meta.post_id
INNER JOIN tags ON tags.id = tags_meta.tag_id
WHERE tags.name = "elizabeth"
ORDER BY videogame.reg_date
```

このコードは、そのタグ "#elizabeth"にするすべてのをすことができます

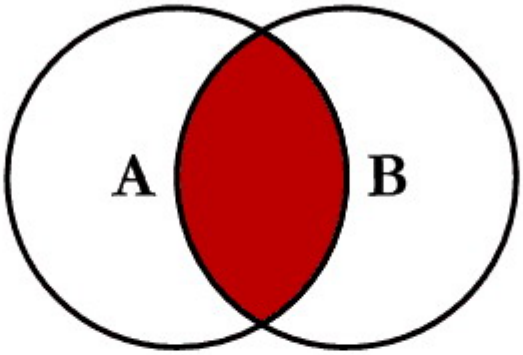
された

あなたがのであれば、このVennダイアグラムは、MySQLにするさまざまなの`JOIN`をするのにちます。

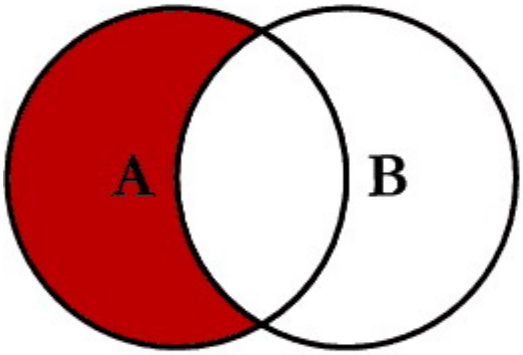
SQL JOINS



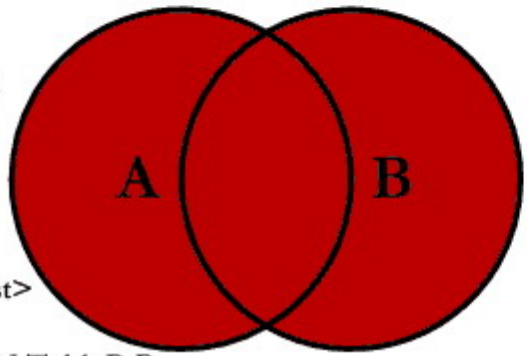

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

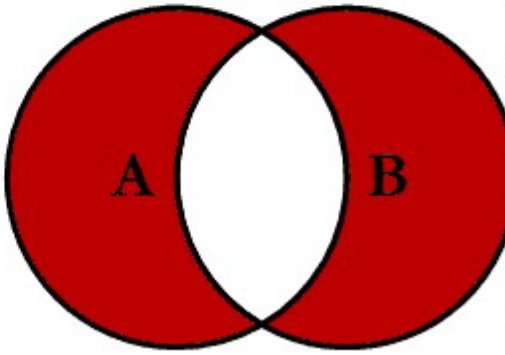



```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```






```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```


オンラインでをむ https://riptutorial.com/ja/mysql/topic/2736/

72:

レプリケーションは、1つのMySQLデータベースサーバーから1つのMySQLデータベースサーバーに[バックアップ]データをコピーするためにされます。

Master - コピーするデータをしているMySQLデータベースサーバ

スレーブ - MySQLデータベースサーバは、マスタによってされるデータをコピーします。

MySQLでは、レプリケーションはデフォルトではです。これは、マスターからをけるためにスレーブをにするがないことをします。たとえば、スレーブがオフになっている、またはマスターにされておらず、でスレーブをオンにりえる、またはマスターにしているは、にマスターとします。

にじて、すべてのデータベース、したデータベース、またはデータベースのされたテーブルをすることができます。

レプリケーションフォーマット

フォーマットには2つのコアタイプがあります

Statement Based ReplicationSBR - SQLをします。この、マスターはSQLをバイナリログにきみます。スレーブへのマスタのレプリケーションは、スレーブでそのSQLをすることによってします。

Row Based ReplicationRBR - されたのみをします。この、マスターは、々ののがどのようにされるかをすイベントをバイナリー·ログにきみます。スレーブへのマスタのレプリケーションは、テーブルへのをすイベントをスレーブにコピーすることによってします。

3の、 *Mixed Based ReplicationMBR*をすることもできます。この、ステートメントベースとロウベースののログがされます。にもしたログがされます。

5.7.7よりいバージョンのMySQLでは、ステートメントベースのがデフォルトでした。 MySQL 5.7.7では、ベースのがデフォルトです。

Examples

マスタ - スレーブレプリケーションセットアップ

セットアップに2つのMySQLサーバをえてみましょう.1つはマスタ、もう1つはスレーブです。

マスターは、されたすべてのアクションのログをするがあることをマスターにします。マスターのログをるべきスレーブサーバをし、マスターのログにがあったでもじことをするがあります。

マスター

まず、マスター—にユーザー—をするがあります。このユーザー—は、スレーブがマスター—とのをするためにされます。

```
CREATE USER 'user_name'@'%' IDENTIFIED BY 'user_password';
GRANT REPLICATION SLAVE ON *.* TO 'user_name'@'%';
FLUSH PRIVILEGES;
```

UsernameとPasswordにって、 `user_name`と`user_password`し`user_name`。

、 `my.inf` Linuxのmy.cnfファイルをするがあります。 [mysqld]セクションにのをめます。

```
server-id = 1
log-bin = mysql-bin.log
binlog-do-db = your_database
```

のは、このMySQLサ—バにIDをりてるためにされます。

2は、されたログファイルにログをきむようにMySQLにします。 Linuxでは、 `log-bin = /home/mysql/logs/mysql-bin.log`ようにできます。レプリケ—ションがすでにされているMySQLサ—バでレプリケ—ションをするは、このディレクトリがすべてのレプリケ—ションログからになっていることをしてください。

3は、ログをきむデ—タベ—スのにされます。 `your_database`をデ—タベ—スできえる`your_database`があります。

`skip-networking`がになっていないことをして、MySQLサ—バをしてくださいマスタ

スレ—ブ

`my.inf`ファイルもスレ—ブでするがあります。 [mysqld]セクションにのをめます。

```
server-id = 2
master-host = master_ip_address
master-connect-retry = 60

master-user = user_name
master-password = user_password
replicate-do-db = your_database

relay-log = slave-relay.log
relay-log-index = slave-relay-log.index
```

のは、このMySQLサ—バにIDをりてるためにされます。このIDはであるがあります。

2は、マスター—サ—バ—のIPアドレスです。マスター—システムのIPにってこれをしてください

3はをでするためにされます。

の2は、スレ—ブにするためにするユ—ザとパスワ—ドをスレ—ブにえます。

のは、するがあるデータベースをします。

の2は、 `relay-log`と`relay-log-index`ファイルのりてにされます。

`skip-networking`がになっていないことをし、MySQLサーバをしてくださいスレーブ

データをスレーブにコピーする

データがにマスタにされる、することはできないように、マスタのすべてのデータベースアクセスをするがあります。これは、Masterでのステートメントをすることでできます。

```
FLUSH TABLES WITH READ LOCK;
```

サーバーにデータがされていないは、のをスキップできます。

たちは、 `mysqldump`をってマスターのデータバックアップをるつもりです

```
mysqldump your_database -u root -p > D://Backup/backup.sql;
```

にって`your_database`とバックアップディレクトリをします。 `backup.sql`れたに`backup.sql`というファイルがされます。

スレーブにデータベースがしないは、のコマンドをしてデータベースをします

```
CREATE DATABASE `your_database`;
```

これで、スレーブMySQLサーバにバックアップをインポートするがあります。

```
mysql -u root -p your_database  <D://Backup/backup.sql
--->Change `your_database` and backup directory according to your setup
```

レプリケーションをする

レプリケーションをするには、マスターのログファイルとログのをするがあります。だから、マスターでのようにする

```
SHOW MASTER STATUS;
```

これにより、のようながられます

```
+--------------------+----------+--------------------------------+------------------+
| File               | Position | Binlog_Do_DB                   | Binlog_Ignore_DB |
+--------------------+----------+--------------------------------+------------------+
| mysql-bin.000001   | 130      | your_database                  |                  |
+--------------------+----------+--------------------------------+------------------+
```

に、スレーブでのコマンドをします。

```
SLAVE STOP;
CHANGE MASTER TO MASTER_HOST='master_ip_address', MASTER_USER='user_name',
   MASTER_PASSWORD='user_password', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=130;
SLAVE START;
```

まずはスレ―ブをめる。に、マスター―ログファイルをどこにするかをにえます。 `MASTER_LOG_FILE` と`MASTER_LOG_POS`については、マスター―で`SHOW MASTER STATUS`コマンドをしてたをしてください。

`MASTER_HOST`でマスター―のIPをし、それにじてユ―ザ―とパスワ―ドをするがあります。

スレ―ブはしています。スレ―ブのは、のコマンドをするとされます

```
SHOW SLAVE STATUS;
```

にマスタで`FLUSH TABLES WITH READ LOCK`をしたは、のコマンドをしてロックからテ―ブルをします

```
UNLOCK TABLES;
```

これでマスター―は、されたすべてのアクションのログをし、スレ―ブサ―バ―はマスター―のログをします。マスター―のログにがするたびに、スレ―ブはそれをします。

レプリケ―ションエラ―

スレ―ブでクエリをにエラ―がすると、MySQLはにレプリケ―ションをしてをし、します。これは、イベントがキ―をきこしたか、がつからず、またはできないことがなです。これがされないでも、このようなエラ―はスキップできます

スレ―ブをハングしているクエリを1つだけスキップするには、のをします

```
SET GLOBAL sql_slave_skip_counter = N;
```

このステ―トメントは、マスター―からのNイベントをスキップします。このステ―トメントは、スレ―ブスレッドがされていないにのみです。それのは、エラ―がします。

```
STOP SLAVE;
SET GLOBAL sql_slave_skip_counter=1;
START SLAVE;
```

によってはこれはありません。しかし、ステ―トメントがステ―トメント・トランザクションのである、エラ―ステ―トメントをスキップするとトランザクションがスキップされるため、ステ―トメントはよりになります。

じエラ―コ―ドをするよりくのクエリをスキップしたい、それらのエラ―をスキップしてスレ―ブがしないようにして、それらをすべてスキップしたいは、 `my.cnf`エラ―コ―ドをスキップするをします。

たとえば、しているエラ―をすべてスキップしたいがあります

```
1062 | Error 'Duplicate entry 'xyz' for key 1' on query
```

に、あなたの`my.cnf`をします

```
slave-skip-errors = 1062
```

のタイプのエラーまたはすべてのエラーコードもスキップできますが、それらのエラーをスキップしてもスレーブがしないようにしてください。とはのとおりです

```
slave-skip-errors=[err_code1,err_code2,...|all]

slave-skip-errors=1062,1053
slave-skip-errors=all
slave-skip-errors=ddl_exist_errors
```

オンラインでをむ https://riptutorial.com/ja/mysql/topic/7218/

73:

- UNION DISTINCT - SELECTをしたの
- UNION ALL - より
- UNION - デフォルトはDISTINCTです。
- SELECT ... UNION SELECT ... - OKですが、 `ORDER BY`ではあいまいです
- SELECT ...UNIONSELECT ...ORDER BY ... - あいまいさをする

UNIONはのCPUをしません。

UNIONはに*をするためのをみます。 * 5.7.3 / MariaDB 10.1、UNIONのいくつかのは、tmpテーブルをせずにをしますしたがってくなります。

Examples

SELECTステートメントをUNIONとみわせる

じの2つのクエリのを`UNION`キーワードとみわせることができます。

たとえば、 `authors` `editors`と`editors` 2つの々のテーブルすべてののリストがなは、のように`UNION`キーワードをできます。

```
select name, email, phone_number
from authors

union

select name, email, phone_number
from editors
```

`union`をでると、がりかれます。せにをすがあるは、のように`ALL`キーワードをできます`UNION ALL`。

ORDER BY

UNIONのをソートするがあるは、のパターンをします。

```
( SELECT ... )
UNION
( SELECT ... )
ORDER BY
```

がなければ、のORDER BYはのSELECTにします。

OFFSETによるページネーション

LIMITをUNIONにするは、これをするパターンです。

```
( SELECT ... ORDER BY x  LIMIT 10 )
UNION
( SELECT ... ORDER BY x  LIMIT 10 )
ORDER BY x  LIMIT 10
```

"10"はどのSELECTからるのかをできないので、それぞれから10をし、さらに`ORDER BY`と`LIMIT`をりしてリストをろすがあります。

10のうち4ページには、このパターンがです。

```
( SELECT ... ORDER BY x  LIMIT 40 )
UNION
( SELECT ... ORDER BY x  LIMIT 40 )
ORDER BY x  LIMIT 30, 10
```

つまり、`SELECT`で4ページのをし、`UNION OFFSET`をします。

なるとのデータの

```
SELECT name, caption as title, year, pages FROM books
UNION
SELECT name, title, year, 0 as pages FROM movies
```

2つのレコードセットをなるにすると、しているレコードセットをデフォルトでエミュレートします。

UNION ALLおよびUNION

SELECT 1,22,44 UNION SELECT 2,33,55

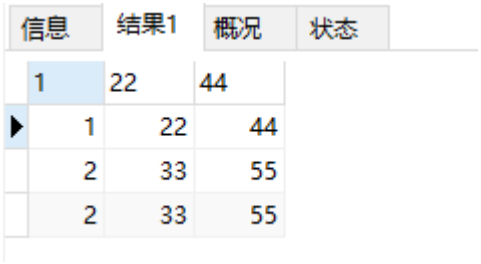
SELECT 1,22,44 UNION SELECT 2,33,55 UNION SELECT 2,33,55

はとじです。

UNION ALLをする

いつ

SELECT 1,22,44 UNION SELECT 2,33,55 UNION ALL SELECT 2,33,55



じをつなるMySQLテーブルのデータをのとクエリにしてマージする

このUNION ALLは、ののデータをし、せにするのとしてします。

```
SELECT YEAR(date_time_column), MONTH(date_time_column), MIN(DATE(date_time_column)),
MAX(DATE(date_time_column)), COUNT(DISTINCT (ip)), COUNT(ip), (COUNT(ip) / COUNT(DISTINCT
(ip))) AS Ratio
FROM (
    (SELECT date_time_column, ip FROM server_log_1 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log_2 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log_3 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log WHERE state = 'action' AND log_id = 150)
) AS table_all
GROUP BY YEAR(date_time_column), MONTH(date_time_column);
```

オンラインでをむ https://riptutorial.com/ja/mysql/topic/3847/

クレジット

S. No		Contributors
1	MySQLをいめる	A. Raza, Aman Dhanda, Andy, Athafoud, CodeWarrior, Community, Confiqure, Dipen Shah, e4c5, Epodax, Giacomo Garabello, greatwolf, inetphantom, JayRizzo, juergen d, Lahiru Ashan, Lambda Ninja, Magisch, Marek Skiba, Md. Nahiduzzaman Rose, moopet, msohng, Noah van der Aa, O. Jones, OverCoder, Panda, Parth Patel, rap-2-h, rhavendc, Romain Vincent, YCF_L
2	1	falsefive
3	Docker-ComposeでMysqlコンテナをインストールする	Marc Alff, molavec
4	ENUM	Philipp, Rick James
5	GRANTによるセキュリティ	Rick James
6	JOINSidとじのテーブルを3つします。	FMashiro
7	JSON	A. Raza, Ben, Drew, e4c5, Manatax, Mark Amery, MohaMad, phatfingers, Rick James, sunkuet02
8	JSONからをする	MohaMad
9	LOAD DATA INFILE	aries12, Asaph, bhrached, CGritton, e4c5, RamenChef, Rick James, WAF
10	MyISAMエンジン	Rick James
11	MyISAMからInnoDBへの	Ponnarasu, Rick James, yukoff
12	MySQL 5.7+のデフォルトのrootパスワードをしてリセットする	Lahiru, ParthaSen
13	mysqldumpをったバックアップ	agold, Asaph, Barranka, Batsu, KalenGi, Mark Amery, Matthew, mnoronha, Ponnarasu, RamenChef, Rick James, still_learning, strangeqargo, Sumit Gupta, Timothy, WAF

14	mysqlimport	Batsu
15	MySQLクライアント	Batsu, Nathaniel Ford, Rick James
16	MySQLのパフォーマンスのヒント	arushi, RamenChef, Rick James, Rodrigo Darti da Costa
17	MySQLのユニオン	Ani Menon, Rick James
18	MySQLのロックテーブル	Ponnarasu, Rick James, vijeeshin
19	MySQL	Florian Genser, Matas Vaitkevicius, RationalDev, Rick James
20	ORDER BY	Florian Genser, Rick James
21	Prepared StatementをしたUn-Pivotテーブル	rpd
22	PREPAREステートメント	kolunar, Rick James, winter
23	PS1をカスタマイズする	Eugene, Wenzhong
24	SSLのセットアップ	4444, a coder, Eugene
25	VIEW	Abhishek Aggrawal, Divya, e4c5, Marina K., Nikita Kurtin, Ponnarasu, R.K123, ratchet, Rick James, WAF, Yury Fedorov, Илья Плотников
26	イベント	Drew, rene
27	インサート	0x49D1, AbcAeffchen, Abubakkar, Aukhan, CGritton, Dinidu, Dreamer, Drew, e4c5, fnkr, gabe3886, Horen, Hugo Buff, Ian Kenney, Johan, Magisch, NEER, Parth Patel, Philipp, Rick James, Riho, strangeqargo, Thuta Aung, zeppelin
28	インデックスとキー	Alex Recarey, Barranka, Ben Visness, Drew, kolunar, Rick James, Sanjeev kumar
29	エラー1055 ONLY_FULL_GROUP_BYかがGROUP BYにない...	Damian Yerrick, O. Jones
30	エラーコード	Drew, e4c5, juergen d, Lucas Paolillo, O. Jones, Ponnarasu, Rick James, WAF, Wojciech Kazior
31	キャラクタセットと	frlan, Rick, Rick James

32	クラスタリング	Drew, Rick James
33	グループする	Adam, Filipe Martins, Lijo, Rick James, Thuta Aung, WAF, whrrgarbl
34	コメントMysql	Franck Dernoncourt, Rick James, WAF, YCF_L
35	サーバー	FMashiro
36	さまざまなプログラミングをしたUTF-8との。	Epodax, Rick James
37	ストアドルーチンきおよび	Abhishek Aggrawal, Abubakkar, Darwin von Corax, Dinidu, Drew, e4c5, juergen d, kolunar, llanato, Rick James, userlond
38	スパースまたはデータの	Batsu, Nate Vaughan
39	セレクト	Ani Menon, Asjad Athick, Benvorth, Bhavin Solanki, Chip, Drew, greatwolf, Inzimam Tariq IT, julienc, KartikKannapur, Kruti Patel, Matthis Kohli, O. Jones, Ponnarasu, Rick James, SeeuD1, ThisIsImpossible, timmyRS, YCF_L, ypercube
40	タイムゾーンの	O. Jones
41	データベースの	Daniel Käfer, Drew, Ponnarasu, R.K123, Rick James, still_learning
42	データ	Batsu, dakab, Drew, Dylan Vander Berg, e4c5, juergen d, MohaMad, Richard Hamilton, Rick James
43	テーブル	4444, Alex Shesterov, alex9311, andygeers, Aryo, Asaph, Barranka, Benvorth, Brad Larson, CPHPython, Darwin von Corax, Dinidu, Drew, fedorqui, HCarrasko, Jean Vitor, John M, Matt, Misa Lazovic, Panda, Parth Patel, Paulo Freitas, Přemysl Šťastný, Rick, Rick James, Ronnie Wang, Saroj Sasmal, Sebastian Brosch, skytreader, Stefan Rogin , Strawberry, Timothy, ultrajohn, user6655061, vijaykumar, Vini.g.fer, Vladimir Kovpak, WAF, YCF_L, Yury Fedorov
44	トランザクション	Ponnarasu, Rick James
45	トリガー	Blag, e4c5, Matas Vaitkevicius, ratchet, WAF, YCF_L
46	ドロップテーブル	Noah van der Aa, Parth Patel, Ponnarasu, R.K123, Rick James, trf, Tushar patel, YCF_L
47	ヌル	Rick James, Sumit Gupta

48	パーティショニング	Majid, Rick James
49	パスワードをする	e4c5, Hardik Kanjariya ツ, Rick James, Viktor, ydaetskcoR
50	バックティック	Drew, SuperDJ
51	ピボットクエリ	Barranka
52	リミットとオフセット	Alvaro Flaño Larrondo, Ani Menon, animuson, ChaoticTwist, Chris Rasys, CPHPython, Ian Gregory, Matt S, Rick James, Sumit Gupta, WAF
53	ログファイル	Drew, Rick James
54	テーブル	Ponnarasu, Rick James
55		juergen d, user2314737
56	の	e4c5, JohnLBevan, kolunar, LiuYan , Matas Vaitkevicius, mayojava, Rick James, Steve Chambers, Thuta Aung, WAF, YCF_L
57		O. Jones
58		Batsu, Drew, e4c5, ForguesR, gabe3886, Khurram, Parth Patel, Ponnarasu, Rick James, strangeqargo, WAF, whrrgarbl, ypercube, Илья Плотников
59	の	kolunar, user6655061
60	マッピングテーブル	Rick James
61		e4c5, RamenChef, Rick James
62		Abubakkar, Batsu, juergen d, kolunar, Rick James, uruloke , WAF
63	しいユーザーをする	Aminadav, Batsu, Hardik Kanjariya ツ, josinalvo, Rick James, WAF
64	との	Abhishek Aggrawal, Drew, Matt S, O. Jones, Rick James, Sumit Gupta
65		4thfloorstudios, Chris, Drew, Khurram, Ponnarasu, Rick James, Sevle
66	とチューニング	ChintaMoney, CodeWarrior, Epodax, Eugene, jan_kiran, Rick James
67		user2314737, YCF_L

68	のをつ	O. Jones
69		Barranka, Dinidu, Drew, JonMark Perry, O. Jones, RamenChef, Richard Hamilton, Rick James
70	したルートパスワードからする	BacLuc, Jen R
71		Artisan72, Batsu, Benvorth, Bikash P, Drew, Matt, Philipp, Rick, Rick James, user3617558
72		Ponnarasu
73		Mattew Whitt, Rick James, Riho, Tarik, wangengzheng