

(Aurora) MySQL Tips

조그마한 팁들

Jongwony

- Kreditjob Developer
 - Wantedlab Data Engineer
 - AWSKRUG CLI Organizer
-
- GitHub: <https://github.com/jongwony>
 - Blog: <http://tech.jongwony.com/>



들어가기 앞서...

- MySQL 5.6, 5.7을 주 대상으로 합니다. (≥ 8.0 의 특징은 다루지 않았습니다.)

→ `docker run --name test-mysql -e MYSQL_ROOT_PASSWORD=root -p 3306:3306 -d mysql:5.7`

- Aurora 를 사용하는 비중이 크지는 않습니다. MySQL 자체에 포커스를 맞추었습니다.
- Oracle, Postgres, SQL Server 등에 해당이 안되는 부분이 있을 수 있습니다.

Index

- Temporary Table
- LOAD DATA
- Character set
- Collation
- Jupyter magic
- Last Insert ID
- Etc.

Temporary Table

세션이 유지되는 동안 사용하는 테이블
세션이 종료되면 테이블 DROP

Temporary Table

세션이 서로 다르다면 충돌의 염려가 없다.
세션을 하나로 이어서 쓰는 서버가 사용하기는 힘들다.

권한이 필요합니다!

```
grant create temporary tables on *.* to 'user'@'%'
```

Temporary Table

일반 테이블 처럼 인덱스를 걸 수 있다?

<https://dev.mysql.com/doc/refman/5.7/en/create-table.html>

13.1.18 CREATE TABLE Syntax

13.1.18.1 CREATE TABLE Statement Retention

13.1.18.2 Files Created by CREATE TABLE

13.1.18.3 CREATE TEMPORARY TABLE Syntax

13.1.18.4 CREATE TABLE ... LIKE Syntax

13.1.18.5 CREATE TABLE ... SELECT Syntax


13.1.18.6 Using FOREIGN KEY Constraints

13.1.18.7 Silent Column Specification Changes

13.1.18.8 CREATE TABLE and Generated Columns

13.1.18.9 Secondary Indexes and Generated Columns

13.1.18.10 Setting NDB_TABLE Options



```
1 CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
2     (create_definition, ...)
3     [table_options]
4     [partition_options]
```

Temporary Table

1. 여러 인덱스 걸기

```
CREATE TEMPORARY TABLE core.my_tmp_table
(
    INDEX my_index_name (tag, time),
    UNIQUE my_unique_index_name (order_number)
)
SELECT *
FROM core.my_big_table
WHERE my_val = 1
```

<https://stackoverflow.com/questions/14397785/create-a-temporary-table-in-mysql-with-an-index-from-a-select>

Temporary Table

2. Primary Key 걸기

```
CREATE TEMPORARY TABLE core.my_tmp_table  
(  
    PRIMARY KEY my_pk (order_number),  
    INDEX idx_key (user_id, time)  
)  
SELECT *  
FROM core.my_big_table
```

<https://stackoverflow.com/questions/14397785/create-a-temporary-table-in-mysql-with-an-index-from-a-select>

Temporary Table

3. 새로운 컬럼 추가

```
CREATE TEMPORARY TABLE core.my_tmp_table
(
    new_id BIGINT NOT NULL AUTO_INCREMENT,
    create_time DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    update_time DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY my_pk (my_new_id),
    INDEX my_unique_index_name (invoice_number)
)
SELECT *
FROM core.my_big_table
```

<https://stackoverflow.com/questions/14397785/create-a-temporary-table-in-mysql-with-an-index-from-a-select>

Temporary Table

4. 기존 컬럼 재정의

```
CREATE TEMPORARY TABLE core.my_tmp_table
(
    tag      BIGINT,
    my_time DATETIME,
    INDEX my_unique_index_name (tag)
)
SELECT *
FROM core.my_big_table
```

<https://stackoverflow.com/questions/14397785/create-a-temporary-table-in-mysql-with-an-index-from-a-select>

Temporary Table

5. 엔진, 캐릭터셋 정의

```
CREATE TEMPORARY TABLE core.my_tmp_table
(
    id      INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    value   BIGINT UNSIGNED NOT NULL DEFAULT 0 UNIQUE,
    location VARCHAR(20)      DEFAULT 'NEEDS TO BE SET',
    country CHAR(2)           DEFAULT 'XX' COMMENT 'Two-letter code',
    INDEX my_index_name (location)
)
ENGINE = InnoDB
SELECT *
FROM core.my_big_table
```

<https://stackoverflow.com/questions/14397785/create-a-temporary-table-in-mysql-with-an-index-from-a-select>

Temporary Table

근데 이러면 테이블 만드는 과정이 추가되는거라 손해 아닙니까?

Temporary Table

- Evaluation of UNION statements.
- Evaluation of some views, such those that use the TEMPTABLE algorithm, UNION, or aggregation.
- Evaluation of derived tables (see [Section 13.2.10.8, “Derived Tables”](#)).
- Tables created for subquery or semijoin materialization (see [Section 8.2.2, “Optimizing Subqueries and Derived Tables”](#)).
- Evaluation of statements that contain an ORDER BY clause and a different GROUP BY clause, or for which the ORDER BY or GROUP BY contains columns from tables other than the first table in the join queue.
- Evaluation of DISTINCT combined with ORDER BY may require a temporary table.
- For queries that use the SQL_SMALL_RESULT modifier, MySQL uses an in-memory temporary table, unless the query also contains elements (described later) that require on-disk storage.
- To evaluate INSERT ... SELECT statements that select from and insert into the same table, MySQL creates an internal temporary table to hold the rows from the SELECT, then inserts those rows into the target table. See [Section 13.2.5.1, “INSERT ... SELECT Syntax”](#).
- Evaluation of multiple-table UPDATE statements.
- Evaluation of GROUP_CONCAT () or COUNT (DISTINCT) expressions.

내부적으로 다양한 케이스를 통해
Temporary Table을 생성합니다.

Union, Subquery, 정렬의 경우도!

<https://dev.mysql.com/doc/refman/5.6/en/internal-temporary-tables.html>

Temporary Table

기존 테이블과 같은 이름으로 만들 수 있다?

이 경우 기존 테이블에 입혀 저서 세션이 끊어질 때 까지 기존 테이블의 스냅샷 테이블이 됩니다.

Temporary Table

LOAD DATA 문과 사용하면 **벌크데이터를 손쉽게 처리**할 수 있습니다.

파이프라인 및 자잘한 작업에 매우 좋습니다.

테스트 및 트랜잭션과 연관된 작업에도 사용하기 좋습니다.

Temporary Table

메모리 테이블과 같은 말이 아니었다?

Temporary table 은 크기가 작은경우 메모리에 작성될 수 있고 그렇지 않으면 디스크에 작성됩니다.

Temporary Table

`tmp_table_size, max_heap_table_size`
이 두 설정값 중 작은값으로 정의됩니다.

이 값보다 작더라도 관련된 테이블에
TEXT, BLOB 타입이 있다면 디스크에 작성됩니다.

Temporary Table



Temporary Table 은 self join 및 재 참조가 되지 않습니다.

```
mysql> select (select * from test) from (select * from test) t;  
ERROR 1137 (HY000): Can't reopen table: 'test'
```

```
mysql> select (select 1 from test) t1, (select 2 from test) t2 from dual;  
ERROR 1137 (HY000): Can't reopen table: 'test'
```

그래서 보통 ORM 과 사용하기는 힘들며 서로다른 temporary table 을 따로 만들어야 합니다. 😓

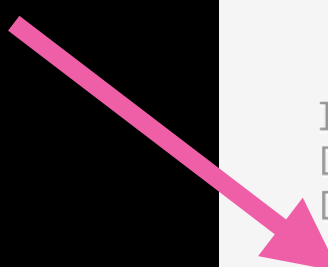
LOAD DATA

텍스트 파일을 읽는 가장 빠른 방법

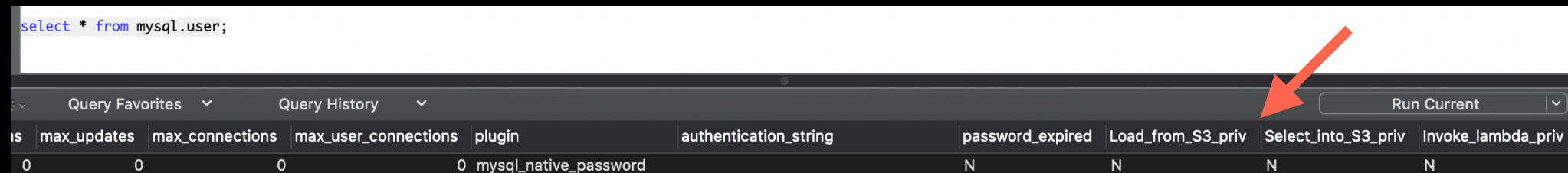
LOAD DATA

- 다양한 케이스의 CSV Handling 가능!
- IF 문 함께 사용 가능!

```
LOAD DATA
  [LOW_PRIORITY | CONCURRENT] [LOCAL]
  INFILE 'file_name'
  [REPLACE | IGNORE]
  INTO TABLE tbl_name
  [PARTITION (partition_name [, partition_name] ...)]
  [CHARACTER SET charset_name]
  [{FIELDS | COLUMNS}
    [TERMINATED BY 'string'
    [[OPTIONALLY] ENCLOSED BY 'char'
    [ESCAPED BY 'char']]
  ]
  [LINES
    [STARTING BY 'string'
    [TERMINATED BY 'string']]
  ]
  [IGNORE number {LINES | ROWS}]
  [(col_name_or_user_var
    [, col_name_or_user_var] ...)]
  [SET col_name=expr | DEFAULT,
    [, col_name=expr | DEFAULT] ...]
```



LOAD DATA




```
select * from mysql.user;
```

	max_updates	max_connections	max_user_connections	plugin	authentication_string	password_expired	Load_from_S3_priv	Select_into_S3_priv	Invoke_lambda_priv
mysql.user	0	0	0	mysql_native_password		N	N	N	N

https://docs.aws.amazon.com/ko_kr/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Integrating.LoadFromS3.html

Aurora S3 export / import 가능!

압축 파일이 지원된다면 좋을 텐데...



```
LOAD DATA FROM S3 [FILE | PREFIX | MANIFEST] 'S3-URI'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[PARTITION (partition_name,...)]
[CHARACTER SET charset_name]
[{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']
]
[LINES
  [STARTING BY 'string']
  [TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var,...)]
[SET col_name = expr,...]
```

Character set

Unicode? Character set? Encoding?

Unicode가 Character set 이고
UTF-8 은 encoding 방식입니다.

utf8(mb3): 3 byte UTF-8 Encoding character set 인 것이죠!

<https://dev.mysql.com/doc/refman/5.7/en/charset-unicode-utf8mb4.html>

Character set

MySQL의 utf8mb4는 utf8 의 super set 입니다.

아래와 같은 유니코드에 확장으로 등록된 한자나
이모지 같은 경우에는 utf8 캐릭터셋을 가진 데이터베이스에 적합하지 않습니다.

```
mysql> insert into test select '😎';  
ERROR 1366 (HY000): Incorrect string value: '\xF0\x9F\x98\x8E' for column 't1' at row 1
```

INSERT 가 되는 경우가 있는데 *show warnings* 에서 같은 메시지가 나타나고 ?로 표시됩니다.

https://charset.fandom.com/ko/wiki/JIS_X_0213_확장_한자_임시_위치

Character set

Column의 Character set 을 확인하세요!

```
create database test_mb3 default character set utf8;
use test_mb3;
create table test select '寬' as t1;
```

```
mysql> show create table test\G
***** 1. row *****
      Table: test
Create Table: CREATE TABLE `test` (
  `t1` varchar(1) CHARACTER SET utf8mb4 NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.01 sec)
```

Character set

connection 에서 기본 **character set** 설정이 되어 있을 수 있습니다.

```
mysql> show variables like 'character%';
```

Variable_name	Value
character_set_client	utf8mb4
character_set_connection	utf8mb4
character_set_database	utf8
character_set_filesystem	binary
character_set_results	utf8mb4
character_set_server	latin1
character_set_system	utf8
character_sets_dir	/usr/share/mysql/charsets/

```
8 rows in set (0.02 sec)
```

Collation

Character set을 비교 및 정렬하기 위한 룰

Collation

이게 JOIN 된다고?

```
mysql> select * from test1 left join test2 on t1 = t2;
```

```
+-----+-----+  
| t1    | t2    |  
+-----+-----+  
| bBBbB | BbBbB | ???  
+-----+-----+  
1 row in set (0.00 sec)
```

```
mysql> select * from test1 left join test2 on t1 collate utf8mb4_bin = t2 collate utf8mb4_bin; -- 2
```

```
+-----+-----+  
| t1    | t2    |  
+-----+-----+  
| bBBbB | NULL  |  
+-----+-----+  
1 row in set (0.00 sec)
```

```
use test;  
create table test1 as  
select 'bBBbB' t1;  
create table test2 as  
select 'BbBbB' t2;
```

Collation

기본적으로 유니코드 캐릭터셋을 정렬하고 비교하기 위해
utf8(mb4)_general_ci collation 을 사용합니다.

↑
case_insensitive임

대부분 MySQL에서 Primary Key를 정수로 하고 Key 가 아닌 경우 Join 을 하는 경우가 많지 않아
이런일이 드물게 일어날 수 있습니다. 그래서 더욱 주의가 필요합니다!

Collation

대소문자 비교가 가능한 Collation 들

```
mysql> show collation where collation not like '%_ci';
```

Collation	Charset	Id	Default	Compiled	Sortlen
big5_bin	big5	84		Yes	1
dec8_bin	dec8	69		Yes	1
cp850_bin	cp850	80		Yes	1
hp8_bin	hp8	72		Yes	1
koi8r_bin	koi8r	74		Yes	1
latin1_bin	latin1	47		Yes	1

Jupyter magic

A 테이블은 RDS1에 있고 B 테이블은 RDS2에 있는데 둘을 조인하고 싶어요.



Jupyter magic

▼ custom

__init__.py

sql_magic.py

```
@magics_class
class SQL(Magics):
    @line_cell_magic
    def load_a(self, line, cell=None):
        return execute_query(cell or line, rdb_reader('ssm_string_A', 'database_A'))

    @line_cell_magic
    def load_b(self, line, cell=None):
        return execute_query(cell or line, rdb_reader('ssm_string_B', 'database_B'))
```

<https://gist.github.com/jongwony/e94a2c89b9df056507a92d6353dfda13>

Jupyter magic

AWS System Manager 로 계정을 관리하고,
custom 모듈을 jupyter notebook에 붙인다면...!

```
%load_ext custom  
  
df1 = %load_a select * from test  
df2 = %load_b select * from test  
  
df = df1.merge(df2, how='left', on='id')
```

<https://gist.github.com/jongwony/e94a2c89b9df056507a92d6353dfda13>

Last Insert ID

INSERT 또는 UPDATE 직 후 Auto Increment 컬럼의 생성값
을 가져오는 방법

Temporary Table 과 같이 세션에 독립적입니다!

Last Insert ID

하지만 여기에 관한 오해가 많습니다.
아래 테이블로 여러 실험을 진행해 보도록 하겠습니다.

```
create table test
(
    id      int not null auto_increment,
    uniq_val varchar(40) default null,
    primary key (id),
    unique key (uniq_val)
);
```

Last Insert ID

Multiple Insert

```
-- multiple insert  
insert into test(uniq_val) values (SHA('1')), (SHA('2')), (SHA('3'));
```

```
mysql> select LAST_INSERT_ID();  
+-----+  
| LAST_INSERT_ID() |  
+-----+  
|                  1 |  
+-----+  
1 row in set (0.00 sec)
```

Last Insert ID

Transaction

```
-- transaction fail
start transaction;
insert into test(uniq_val) values (SHA('4'));
insert into test(uniq_val) values (SHA('5'));
insert into test(uniq_val) values (SHA('6'));
rollback;
select LAST_INSERT_ID()
```

```
mysql> select LAST_INSERT_ID()
-> ;
+-----+
| LAST_INSERT_ID() |
+-----+
|                6 |
+-----+
1 row in set (0.00 sec)
```

Last Insert ID

Return 0

프로그래밍 언어로 **LAST_INSERT_ID()**를 사용했을 때
간혹 0이 반환되는 경우가 있습니다.

Last Insert ID

28.7.7.38 mysql_insert_id()

```
uint64_t mysql_insert_id(MYSQL *mysql)
```

Description

Returns the value generated for an `AUTO_INCREMENT` column by the previous `INSERT` or `UPDATE` statement. Use this function after you have performed an `INSERT` statement into a table that contains an `AUTO_INCREMENT` field, or have used `INSERT` or `UPDATE` to set a column value with `LAST_INSERT_ID(expr)`.

The return value of `mysql_insert_id()` is always zero unless explicitly updated under one of the following conditions:

- `INSERT` statements that store a value into an `AUTO_INCREMENT` column. This is true whether the value is automatically generated by storing the special values `NULL` or `0` into the column, or is an explicit nonspecial value.
- In the case of a multiple-row `INSERT` statement, `mysql_insert_id()` returns the first automatically generated `AUTO_INCREMENT` value that was successfully inserted.

If no rows are successfully inserted, `mysql_insert_id()` returns `0`.

- If an `INSERT ... SELECT` statement is executed, and no automatically generated value is successfully inserted, `mysql_insert_id()` returns the ID of the last inserted row.
- If an `INSERT ... SELECT` statement uses `LAST_INSERT_ID(expr)`, `mysql_insert_id()` returns *expr*.
- `INSERT` statements that generate an `AUTO_INCREMENT` value by inserting `LAST_INSERT_ID(expr)` into any column or by updating any column to `LAST_INSERT_ID(expr)`.
- If the previous statement returned an error, the value of `mysql_insert_id()` is undefined.

Last Insert ID

INSERT ... ON DUPLICATE KEY UPDATE

Python 을 사용한 경우

INSERT 의 경우 제대로 되고 **UPDATE** 시에 **0**이 반환되는 경우가 있습니다...
따라서 아래와 같은 **안전장치**를 사용했습니다.

```
def upsert_last_id(data, engine, table, pk, *,
                  values: list = None, text: str = None):
    """https://stackoverflow.com/questions/778534/mysql-on-duplicate-key-last-insert-id"""
    temp = check_frame(data)
    assert len(temp) == 1
    sql = f'''
INSERT INTO {table}({','.join(map(str, temp.columns))})
VALUES ({','.join(f'%({col})s' for col in temp.columns)})
ON DUPLICATE KEY UPDATE
{f'{{pk}}=LAST_INSERT_ID({{pk}})'}
'''

    if values:
        sql += ', ' + ', '.join(f'{{k}}=VALUES({{k}})' for k in values)
    if text:
        sql += ', ' + text

    # execute must pd.Series type
    execute_insert(sql, engine, temp)
    last_id = engine.execute('SELECT LAST_INSERT_ID()').fetchone()

    return last_id[0] if last_id else None
```


Etc.

라이트닝 토크가 되는 건가?

Etc.

쿼리 스크립트 작성시 시작부분에 **SET @@session.time_zone = 'Asia/Seoul';**

을 추가하시면 **now, current_timestamp** 등의 함수가 모두 한국시간으로 바뀝니다.

Press H

Etc.

Version Control: Alembic(Python)

<https://alembic.sqlalchemy.org/en/latest/>

Etc.

첫 단추의 크기가 절반(?)이다(설계편).

<https://www.red-gate.com/simple-talk/sql/database-administration/ten-common-database-design-mistakes/>
<https://www.red-gate.com/simple-talk/sql/database-administration/five-simple-database-design-errors-you-should-avoid/>

이런 관계형 데이터 설계와 비슷한 양질의 커뮤니티 알고 계시면 소개 부탁드립니다. 😊

Summary

사실 팁도 도움은 되겠지만 관계형 데이터베이스 명칭에 맞게
관계 설계를 잘하는 것이 가장 좋습니다.

References

- [MySQL Docs](#)
- [Books](#)
 - [High Performance MySQL](#)
 - [MySQL CookBook](#)
- [StackOverflow\(중간중간 각종 링크들\)](#)
- <https://www.percona.com/blog/2019/07/17/mysql-disk-space-exhaustion-for-implicit-temporary-tables/>
- [Red Gate Community Blog](#)
- [경험 당시 적어 놓은 블로그, Confluence](#)
 - <http://tech.jongwony.com/>

감사합니다.