

★ 데이터 형식의 종류 및 **내장함수** ★

■ 내장 함수

종 류

제어흐름 함수

날짜/시간 함수

XML 함수

정보 함수

문자열 함수

전체 텍스트 검색 함수

비트 함수

JSON 함수

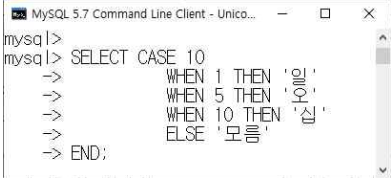
수학 함수

형 변환 함수

보안/압축 함수

대용량 데이터 저장 함수

☞ 제어 흐름 함수

함수	설명(또는 형식)
IF (수식, 참, 거짓)	- 수식이 참이면 두 번째 인수를 반환하고, 거짓이면 세 번째 인수를 반환
IFNULL (수식1, 수식2)	- 수식1이 NULL이 아니면 수식1을 반환하고, 수식1이 NULL이면 수식2를 반환
NULLIF (수식1, 수식2)	- 수식1과 수식2가 같으면 NULL을 반환하고, 다르면 수식1을 반환
CASE ... WHEN ... ELSE ... END	- 다중 분기 

☞ 문자열 함수

함수	설명(또는 형식)
ASCII (아스키코드), CHAR (숫자)	- 문자의 아스키코드 값을 반환하거나 숫자의 아스키코드 값에 해당하는 문자를 반환
BIT_LENGTH (문자열)	- 할당된 비트 크기를 반환
CHAR_LENGTH (문자열)	- 문자의 개수를 반환
LENGTH (문자열)	- 바이트 수를 반환
CONCAT (문자열1, 문자열2, ...)	- 문자열을 이어줌
CONCAT_WS (구분자, 문자열1, 문자열2, ...)	- 구분자와 함께 문자열을 이어줌
ELT (위치, 문자열1, 문자열2, ...)	- 첫 번째 인수인 '위치'에 적힌 숫자를 보고 그 숫자 번째에 있는 문자열을 반환
FIELD (찾을_문자열, 문자열1, 문자열2, ...)	- 찾을 문자열의 위치를 찾아 반환하는데, 매치되는 문자열이 없으면 0을 반환
FIND_IN_SET (찾을_문자열, 문자열_리스트)	- 찾을 문자열을 문자열 리스트에서 찾아 위치를 반환
INSTR (기준_문자열, 부분_문자열)	- 기준 문자열에서 부분 문자열을 찾아가 시작 위치를 반환
LOCATE (부분_문자열, 기준_문자열)	- INSTR() 함수와 동일하지만 파라미터의 순서가 반대

☞ 문자열 함수 (계속...)

함수	설명(또는 형식)
FORMAT (숫자, 소수점_자릿수)	- 숫자를 소수점 이하 자릿수까지 표현하고 1000단위마다 쉼표(,)를 넣음
BIN (숫자), HEX (숫자), OCT (숫자)	- 2진수, 16진수, 8진수의 값을 반환
INSERT (기준_문자열, 위치, 길이, 삽입할_문자열)	- 기준 문자열의 위치부터 길이만큼을 지우고 삽입할 문자열을 끼워넣음
LEFT (문자열, 길이), RIGHT (문자열, 길이)	- 왼쪽 또는 오른쪽에서 문자열의 길이만큼 반환
LOWER (문자열), UPPER (문자열)	- 대문자를 소문자로, 소문자를 대문자로 바꿈
LPAD (문자열, 길이, 채울_문자열), RPAD (문자열, 길이, 채울_문자열)	- 문자열을 길이만큼 늘리고 빈 곳을 채울 문자열로 채움
LTRIM (문자열), RTRIM (문자열)	- 문자열의 왼쪽 또는 오른쪽 공백을 제거 (중간의 공백은 제거되지 않음)
TRIM (문자열) TRIM (방향 자를_문자열 FROM 문자열)	- TRIM () 함수는 문자열의 앞뒤 공백을 모두 없앴 - ‘방향’ 인자에는 앞을 의미하는 LEADING , 양쪽을 의미하는 BOTH , 뒤를 의미하는 TRAILING 이 올 수 있음
REPEAT (문자열, 횟수)	- 문자열을 횟수만큼 반복
REPLACE (문자열, 원래_문자열, 바꿀_문자열)	- 문자열에서 원래 문자열을 찾아 바꿀 문자열로 치환
REVERSE (문자열)	- 문자열의 순서를 거꾸로 반환
SPACE (길이)	- 길이만큼의 공백을 반환
SUBSTRING (문자열, 시작위치, 길이), SUBSTRING (문자열 FROM 시작위치 FOR 길이)	- 시작 위치부터 길이만큼 문자를 반환 - 길이를 생략하면 문자열의 끝까지 반환
SUBSTRING_INDEX (문자열, 구분자, 횟수)	- 문자열에서 구분자가 왼쪽부터 ‘횟수’ 인자에 적힌 숫자 번째 나오면 그 이후의 문자열은 버리고 앞에 있는 문자열만 출력 - 횟수가 음수이면 오른쪽부터 세어 왼쪽의 남은 문자열을 버리고 출력

MEMO

☞ 수학 함수

함수	설명(또는 형식)
ABS (숫자)	- 숫자의 절댓값을 계산
CEILING (숫자), FLOOR (숫자), ROUND (숫자)	- 올림, 내림, 반올림을 계산
CONV (숫자, 원래_진수, 변환할_진수)	- 숫자를 원래 진수에서 변환할 진수로 변환
DEGREES (숫자), RADIANS (숫자), PI ()	- DEGREES () 함수는 라디안 값을 각도 값으로 변환 - RADIANS () 함수는 각도 값을 라디안 값으로 변환 - PI () 함수는 파이 값인 3.141592를 반환
MOD (숫자1, 숫자2), 숫자1 % 숫자2, 숫자1 MOD 숫자2	- 숫자1을 숫자2로 나눈 나머지 값을 반환
POW (숫자1, 숫자2), SQRT (숫자)	- POW () 함수는 숫자1을 숫자2만큼 거듭제곱한 값을 반환 - SQRT () 함수는 숫자의 제곱근을 반환
RAND ()	- 0 이상 1 미만의 실수를 구함
SIGN (숫자)	- 숫자가 양수, 0, 음수인지 구하여 1, 0, -1 중 하나를 반환
TRUNCATE (숫자, 정수)	- 숫자를 소수점을 기준으로 정수 위치까지 구하고 나머지는 버림
ACOS (숫자), ASIN (숫자) ATAN (숫자), ATAN2 (숫자1, 숫자2) SIN (숫자), COS (숫자), TAN (숫자)	- 삼각함수 관련
EXP (X), LN (숫자), LOG (숫자), LOG (밑수, 숫자), LOG2 (숫자), LOG10 (숫자)	- 지수함수, 로그함수 관련

MEMO

☞ 날짜/시간 함수

함수	설명(또는 형식)
ADDDATE (날짜, 차이), SUBDATE (날짜, 차이)	- 날짜를 기준으로 차이를 더하거나 빼 날짜를 반환
ADDTIME (날짜/시간, 시간), SUBTIME (날짜/시간, 시간)	- 날짜/시간을 기준으로 시간을 더하거나 빼 결과를 반환
YEAR (날짜), MONTH (날짜), DAY (날짜), hour (시간), MINUTE (시간), SECOND (시간), MICROSECOND (시간)	- 날짜 또는 시간에서 연, 월, 일, 시, 분, 초, 밀리초를 구함
DATE (), TIME ()	- DATETIME 형식에서 연-월-일과 시:분:초만 추출
DATEDIFF (날짜1, 날짜2), TIMEDIFF (날짜1또는 시간1, 날짜1 또는 시간2)	- DATEDIFF() 함수는 날짜1-날짜2의 결과를 반환
DAYOFWEEK (날짜) MONTHNAME () DAYOFYEAR (날짜)	- DAYOFWEEK() 함수는 요일(1: 일~7: 토) 반환 - MONTHNAME() 함수는 월의 영문(January ~December) 반환 - DAYOFYEAR() 함수는 1년 중 몇 번째 날(1~366)인지를 반환
LAST_DAY (날짜)	- 입력한 월의 마지막 날짜를 반환
MAKEDATE (연도, 정수)	- 연도의 첫날부터 정수만큼 지난 날짜를 반환
MAKETIME (시, 분, 초)	- 시, 분, 초를 이용하여 ‘시:분:초’의 TIME 형식을 만들
PERIOD_ADD (연월, 개월수) PERIOD_DIFF (연월1, 연월2)	- PERIOD_ADD() 함수는 연월부터 개월 수만큼 지난 연월을 반환 - PERIOD_DIFF()는 연월1-연월2의 개월 수를 반환
QUARTER (날짜)	- 날짜가 4분기 중에서 몇 분기인지를 반환
TIME_TO_SEC (시간)	- 시간을 초 단위로 반환
CURDATE () CURTIME () NOW (), SYSDATE ()	- CURDATE()는 현재 ‘연-월-일’ 반환 - CURTIME()은 현재 ‘시:분:초’ 반환 - NOW()와 SYSDATE()는 현재 ‘연-월-일 시:분:초’ 반환

MEMO

☞ 시스템 정보 함수

함수	설명(또는 형식)
USER(), DATABASE()	- 현재 사용자와 현재 선택된 데이터베이스를 반환
FOUND_ROWS()	- 바로 앞의 SELECT 문에서 조회된 행의 개수를 반환
ROW_COUNT()	- 바로 앞의 INSERT, UPDATE, DELETE 문에서 삽입, 수정, 삭제된 행의 개수를 반환
SLEEP(초)	- 쿼리의 실행을 잠깐 멈춤
VERSION()	- 현재 MySQL의 버전을 출력

MEMO

☞ JSON 데이터와 대용량 데이터 저장

- 웹 환경이나 모바일 응용프로그램 등에서 데이터를 교환하기 위해 만든 개방형 표준 포맷
- JSON으로 작성한 데이터는 **속성(key)과 값(value)의 쌍으로 구성**
- JSON은 자바스크립트 언어에서 파생되었지만 특정한 프로그래밍 언어에 종속되지 않은 독립적인 데이터 포맷

	userName	height
▶	강호동	182
	이휘재	180
	남희석	180
	박수홍	183

MySQL 테이블

JSON_OBJECT()
JSON_ARRAY()

```
{ "name": "강호동", "height": 182 }
{ "name": "이휘재", "height": 180 }
{ "name": "남희석", "height": 180 }
{ "name": "박수홍", "height": 183 }
```

JSON 데이터

```
MySQL 5.7 Command Line Client - Unicode
mysql>
mysql> use cookdb;
Database changed
mysql>
mysql> SELECT JSON_OBJECT('name', userName, 'height', height) AS 'JSON 값'
      -> FROM userTBL
      -> WHERE height >= 180;
+-----+
| JSON 값 |
+-----+
| {"name": "강호동", "height": 182} |
| {"name": "이휘재", "height": 180} |
| {"name": "남희석", "height": 180} |
| {"name": "박수홍", "height": 183} |
+-----+
4 rows in set (0.01 sec)

mysql>
```

함수	설명(또는 형식)
JSON_OBJECT() JSON_ARRAY()	- 문자열이 JSON 형식을 만족하면 1을 반환, 그렇지 않으면 0을 반환
JSON_VALID()	- 세 번째 파라미터에 주어진 문자열의 위치를 반환
JSON_SEARCH()	- 두 번째 파라미터에는 one과 all 중 하나가 올 수 있음 - one은 처음으로 매치되는 하나만 반환하고, all은 매치되는 모든 것을 반환
JSON_EXTRACT()	- JSON_SEARCH()와 반대로 지정한 지정된 위치의 값을 반환
JSON_INSERT()	- 새로운 값을 추출
JSON_REPLACE()	- 값을 변경
JSON_REMOVE()	- 지정된 항목을 삭제

```

1 • SET @json='{ "userTBL" :
2   [
3     {"name":"강호동", "height":182},
4     {"name":"이휘재", "height":180},
5     {"name":"남희석", "height":180},
6     {"name":"박수홍", "height":183}
7   ]
8   }';
9 • SELECT JSON_VALID(@json) AS JSON_VALID;
10 • SELECT JSON_SEARCH(@json, 'one', '남희석') AS JSON_SEARCH;
11 • SELECT JSON_EXTRACT(@json, '$.userTBL[2].name') AS JSON_EXTRACT;
12 • SELECT JSON_INSERT(@json, '$.userTBL[0].mDate', '2019-09-09') AS JSON_INSERT;
13 • SELECT JSON_REPLACE(@json, '$.userTBL[0].name', '토마스') AS JSON_REPLACE;
14 • SELECT JSON_REMOVE(@json, '$.userTBL[0]') AS JSON_REMOVE;

```

- 9행: 문자열이 JSON 형식을 만족하면 1을 반환, 그렇지 않으면 0을 반환

```

mysql> SET @json='{ "userTBL" :
>   [
>     {"name":"강호동", "height":182},
>     {"name":"이휘재", "height":180},
>     {"name":"남희석", "height":180},
>     {"name":"박수홍", "height":183}
>   ]
>   }';
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> SELECT JSON_VALID(@json) AS JSON_VALID;
+-----+
| JSON_VALID |
+-----+
|          1 |
+-----+
1 row in set (0.00 sec)

```

- 10행: '남희석'은 userTBL[2]의 name에 해당하는 부분에 위치

```

mysql> SELECT JSON_SEARCH(@json, 'one', '남희석') AS JSON_SEARCH;
+-----+
| JSON_SEARCH |
+-----+
| "$.userTBL[2].name" |
+-----+
1 row in set (0.00 sec)

```


- 11행: JSON_SEARCH()와 반대로 지정한 지정된 위치의 값을 반환

```
MySQL 5.7 Command Line Client - Unicode
mysql> SELECT JSON_EXTRACT(@json, '$.userTBL[2].name') AS JSON_EXTRACT;
+-----+
| JSON_EXTRACT |
+-----+
| "남희석"      |
+-----+
1 row in set (0.00 sec)

mysql>
```

- 12행: 새로운 값을 추출, userTBL[0]에 mDate가 추가됨.

```
MySQL 5.7 Command Line Client - Unicode
mysql> SELECT JSON_INSERT(@json, '$.userTBL[0].mDate', '2019-09-09') AS JSON_INSERT;
+-----+
| JSON_INSERT |
+-----+
| {"userTBL": [{"name": "강호동", "mDate": "2019-09-09", "height": 182}, {"name": "이휘재", "height": 180}, {"name": "남희석", "height": 180}, {"name": "박수홍", "height": 183}]} |
+-----+
1 row in set (0.00 sec)

mysql>
```

- 13행: 값을 변경, userTBL[0]의 name 부분이 '토마스'로 변경됨.

```
MySQL 5.7 Command Line Client - Unicode
mysql> SELECT JSON_REPLACE(@json, '$.userTBL[0].name', '토마스') AS JSON_REPLACE;
+-----+
| JSON_REPLACE |
+-----+
| {"userTBL": [{"name": "토마스", "height": 182}, {"name": "이휘재", "height": 180}, {"name": "남희석", "height": 180}, {"name": "박수홍", "height": 183}]} |
+-----+
1 row in set (0.00 sec)

mysql>
```

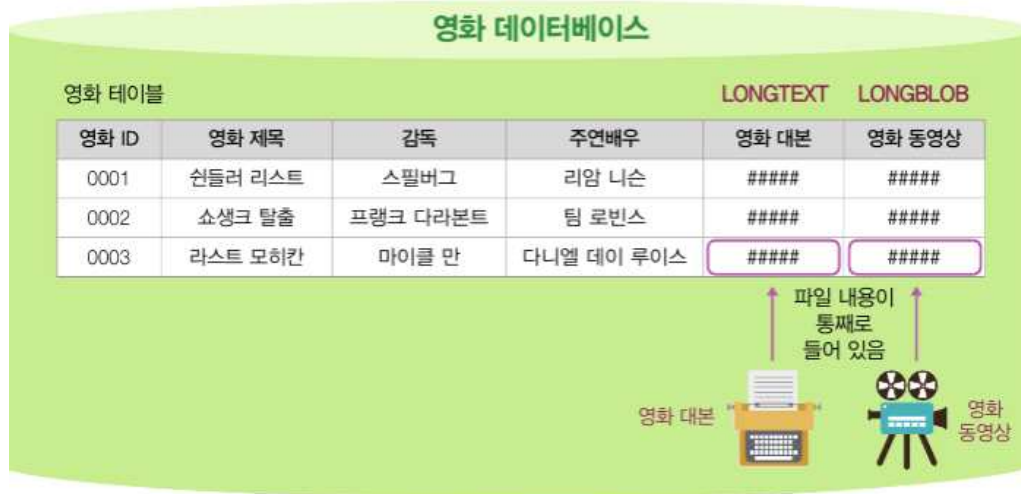
- 14행: 지정된 항목을 삭제, userTBL[0]의 항목이 통째로 삭제됨.

```
MySQL 5.7 Command Line Client - Unicode
mysql> SELECT JSON_REMOVE(@json, '$.userTBL[0]') AS JSON_REMOVE;
+-----+
| JSON_REMOVE |
+-----+
| {"userTBL": [{"name": "이휘재", "height": 180}, {"name": "남희석", "height": 180}, {"name": "박수홍", "height": 183}]} |
+-----+
1 row in set (0.00 sec)

mysql>
```

☞ 대용량 데이터 함수

- MySQL은 대용량 데이터(Large Object, LOB)를 저장하기 위해 LONGTEXT, LONGBLOB 데이터 형식을 지원
- 이 데이터 형식을 이용하면 약 4GB의 파일을 하나의 데이터로 저장할 수 있음.



함수	설명(또는 형식)
LONGTEXT()	- 대용량 텍스트 파일
LONGBLOB()	- 대용량 바이너리 파일
LOAD_FILE()	- 텍스트 파일과 바이너리 파일을 로드 - INSERT INTO movieTBL VALUES (1, '썬들러 리스트', '스필버그', '리암 니슨', LOAD_FILE('C:/Movies/Schindler.txt'), LOAD_FILE('C:/Movies/Schindler.mp4'));

CHECK (my.ini)

mysql> **SHOW** variables **LIKE** 'max_allowed_packet'; -- 용량 체크

```
MySQL 5.7 Command Line Client - Unicode
mysql> SHOW variables LIKE 'max_allowed_packet';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_allowed_packet | 4194304 |
+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql>
```

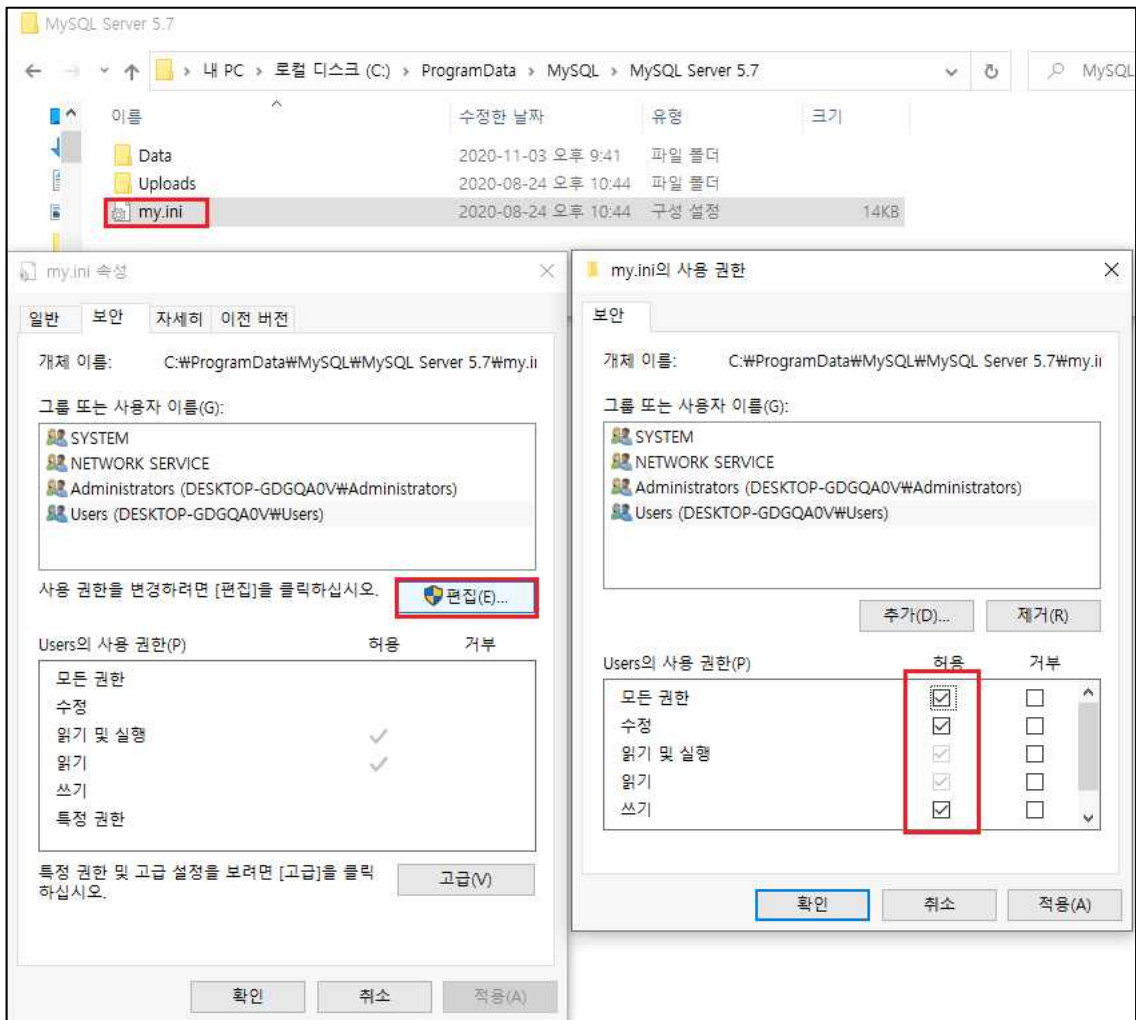
mysql> **SHOW** variables **LIKE** 'secure_file_priv'; -- 파일을 업로드/다운로드 할 폴더 경로

```
MySQL 5.7 Command Line Client - Unicode
mysql> SHOW variables LIKE 'secure_file_priv';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| secure_file_priv | C:\ProgramData\MySQL\MySQL Server 5.7\Uploads\ |
+-----+-----+
1 row in set, 1 warning (0.00 sec)
```


- my.ini 파일 위치: C:\ProgramData\MySQL\MySQL Server 5.7 (버전에 따라 숫자는 다를 수 있음)

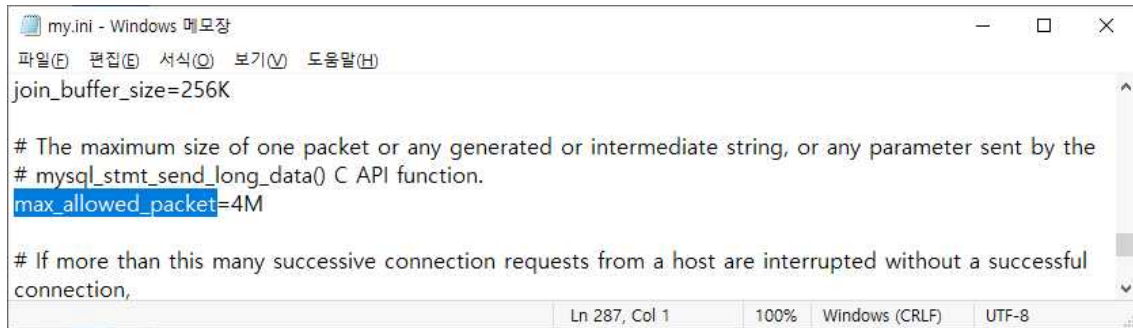


- my.ini 파일을 수정하기 위한 권한 설정



MEMO

- **max_allowed_packet** : 패킷 용량을 4M에서 1G로 수정

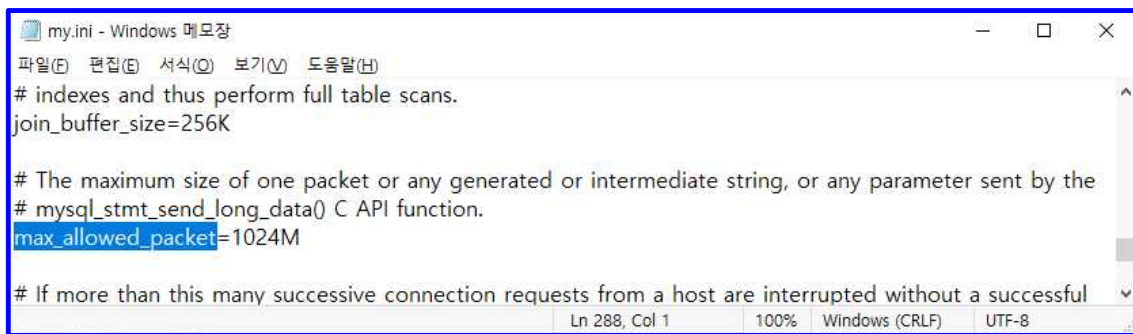


```

my.ini - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
join_buffer_size=256K

# The maximum size of one packet or any generated or intermediate string, or any parameter sent by the
# mysql_stmt_send_long_data() C API function.
max_allowed_packet=4M

# If more than this many successive connection requests from a host are interrupted without a successful
connection,
Ln 287, Col 1 100% Windows (CRLF) UTF-8
  
```



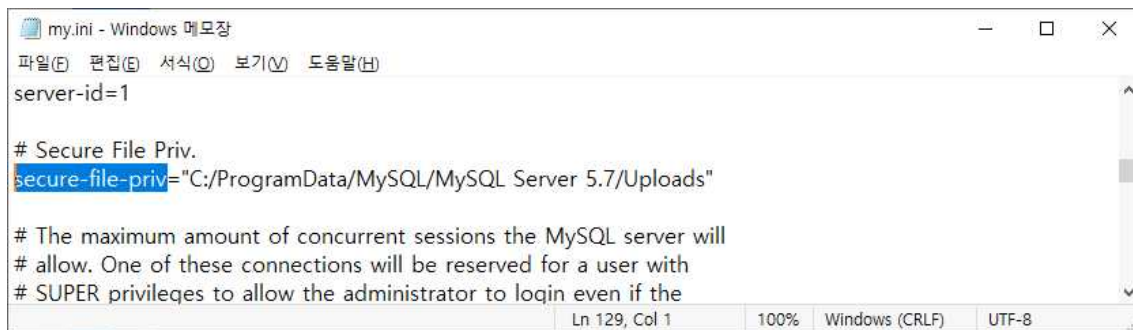
```

my.ini - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
# indexes and thus perform full table scans.
join_buffer_size=256K

# The maximum size of one packet or any generated or intermediate string, or any parameter sent by the
# mysql_stmt_send_long_data() C API function.
max_allowed_packet=1024M

# If more than this many successive connection requests from a host are interrupted without a successful
Ln 288, Col 1 100% Windows (CRLF) UTF-8
  
```

- **secure-file-priv** : 파일을 업로드하거나 다운로드할 폴더 경로를 별도로 추가

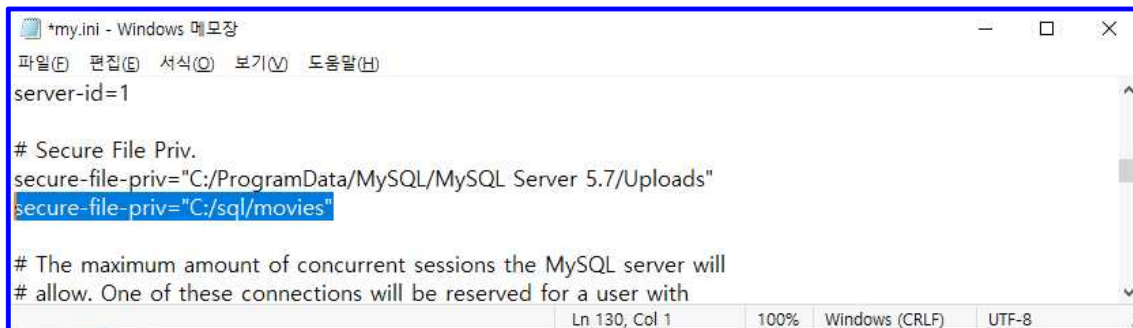


```

my.ini - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
server-id=1

# Secure File Priv.
secure-file-priv="C:/ProgramData/MySQL/MySQL Server 5.7/Uploads"

# The maximum amount of concurrent sessions the MySQL server will
# allow. One of these connections will be reserved for a user with
# SUPER privileges to allow the administrator to login even if the
Ln 129, Col 1 100% Windows (CRLF) UTF-8
  
```



```

*my.ini - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
server-id=1

# Secure File Priv.
secure-file-priv="C:/ProgramData/MySQL/MySQL Server 5.7/Uploads"
secure-file-priv="C:/sql/movies"

# The maximum amount of concurrent sessions the MySQL server will
# allow. One of these connections will be reserved for a user with
Ln 130, Col 1 100% Windows (CRLF) UTF-8
  
```

MEMO

- PowerShell을 이용하여 **my.ini** 파일의 위치를 확인해보자.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\WKihong> cmd
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\WKihong> cd %programdata%
C:\ProgramData> cd mysql
C:\ProgramData\MySQL> cd "mysql server 5.7"
C:\ProgramData\MySQL\MySQL Server 5.7> dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: A46C-2A72

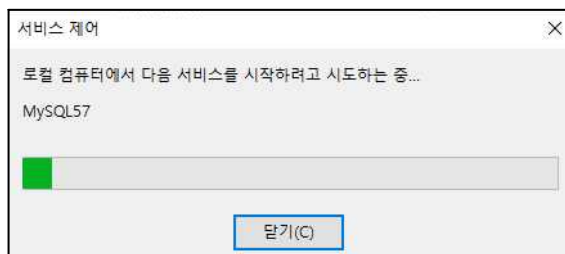
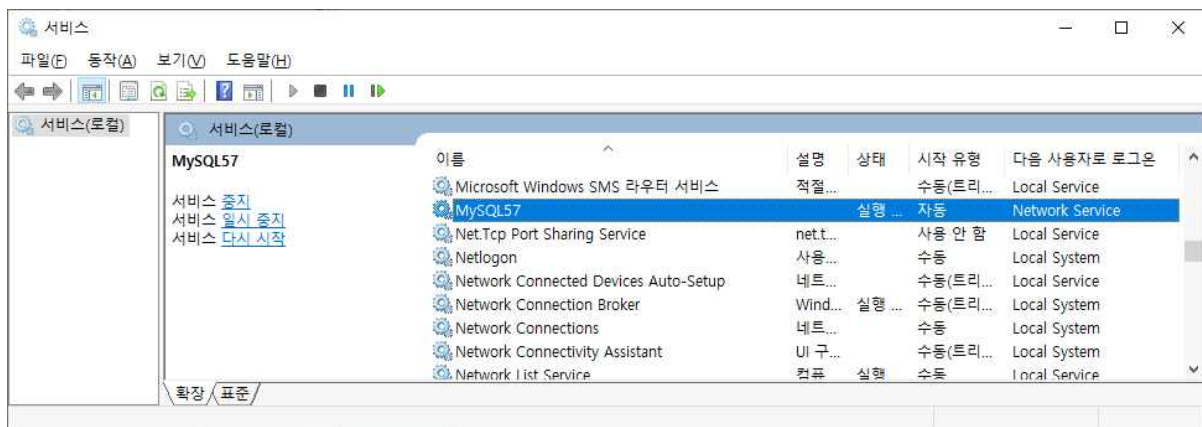
C:\ProgramData\MySQL\MySQL Server 5.7 디렉터리

2020-08-24 오후 10:44 <DIR> .
2020-08-24 오후 10:44 <DIR> ..
2020-11-03 오후 09:41 <DIR> Data
2020-11-05 오전 07:54 13,834 my.ini
2020-08-24 오후 10:44 <DIR> Uploads
1개 파일 13,834 바이트
4개 디렉터리 336,934,846,464 바이트 남음

C:\ProgramData\MySQL\MySQL Server 5.7>

```

- **my.ini** 파일을 수정하면 **MySQL** 서버를 재시작해야 함.



■ 영화 DB / Table 생성, 및 Insert / Select 실습 ■

- 영화를 저장하기 위한 데이터베이스(movieDB)와 테이블(movieTBL)을 만들기
- 테스트를 위한 영화 대본과 mp4 파일은 팀즈 9주차의 파일란에 업로드 해놨습니다.

The screenshot shows the SQL Navigator interface with the following SQL code:

```

1 • CREATE DATABASE movieDB;
2 • USE movieDB;
3
4 • CREATE TABLE movieTBL (
5     movie_id INT,
6     movie_title VARCHAR(30),
7     movie_director VARCHAR(20),
8     movie_star VARCHAR(20),
9     movie_script LONGTEXT,
10    movie_film LONGBLOB )
11    DEFAULT CHARSET=utf8mb4;
  
```

On the right, a file explorer shows the 'movies' directory containing files like Movies.zip, Mohican.mp4, Schindler.mp4, Shawshank.mp4, Mohican.txt, Schindler.txt, and Shawshank.txt.

- movieTBL에 대용량 데이터 Insert한 후, Select 해보기

The screenshot shows the SQL Navigator interface with the following SQL code:

```

1 • INSERT INTO movieTBL VALUES (1, '원들러 리스트', '스필버그', '리암 니슨',
2     LOAD_FILE('C:/sql/movies/Schindler.txt'),
3     LOAD_FILE('C:/sql/movies/Schindler.mp4'));
4
5 • INSERT INTO movieTBL VALUES (2, '쇼생크 탈출', '프랭크 다라본트', '팀 로빈스',
6     LOAD_FILE('C:/sql/movies/Shawshank.txt'),
7     LOAD_FILE('C:/sql/movies/Shawshank.mp4'));
8
9 • INSERT INTO movieTBL VALUES (3, '라스트 모히칸', '마이클 만', '다니엘 데이 루이스',
10    LOAD_FILE('C:/sql/movies/Mohican.txt'),
11    LOAD_FILE('C:/sql/movies/Mohican.mp4'));
  
```

The screenshot shows the Output window with the following execution results:

Time	Action	Message
08:14:05	CREATE DATABASE movieDB	1 row(s) affected
08:14:05	USE movieDB	0 row(s) affected
08:14:05	CREATE TABLE movieTBL (movie_id INT, movie_title VARCHAR(30), movie_director VARCHAR(20), movie_star VARCHAR(20), movie_script LONGTEXT, m...	0 row(s) affected
08:26:45	INSERT INTO movieTBL VALUES (1, '원들러 리스트', '스필버그', '리암 니슨', LOAD_FILE(C:/sql/movies/Schindler.txt), LOAD_FILE(C:/sql/movies/Schindl...	1 row(s) affected
08:26:46	INSERT INTO movieTBL VALUES (2, '쇼생크 탈출', '프랭크 다라본트', '팀 로빈스', LOAD_FILE(C:/sql/movies/Shawshank.txt), LOAD_FILE(C:/sql/movies/...	1 row(s) affected
08:26:46	INSERT INTO movieTBL VALUES (3, '라스트 모히칸', '마이클 만', '다니엘 데이 루이스', LOAD_FILE(C:/sql/movies/Mohican.txt), LOAD_FILE(C:/sql/movie...	1 row(s) affected

The screenshot shows the SQL Navigator interface with the following SQL code:

```

1 • select * from movieTBL;
  
```

Below the code, the Result Grid shows the following data:

movie_id	movie_title	movie_director	movie_star	movie_script	movie_film
1	원들러 리스트	스필버그	리암 니슨	{1939년 9월 독일은 침공 2주만에 폴란드 군을 대파했다. 유대인에게는 가...	BL08
2	쇼생크 탈출	프랭크 다라본트	팀 로빈스	은행원인 앤디 듀플레인은 아내와 프로골퍼인 그의 정부를 죽였다는 누명...	BL08
3	라스트 모히칸	마이클 만	다니엘 데이 루이스	마이클 만Michael Mann이 영화화하려 했던 토머스 페니모어 쿠퍼의 [모히...	BL08

- 영화 대본과 파일(바이너리파일)을 다운받기

```

SQL File 3*  SQL File 4*  SQL File 5*  SQL File 5*
1 • select * from movieTBL;
2
3  -- (영화대본) 데이터를 파일로 다운받기
4 • SELECT movie_script FROM movieTBL WHERE movie_id=1
5     INTO OUTFILE 'C:/sql/movies/Schindler_out.txt'
6     LINES TERMINATED BY '\\n';
7
8  -- (영화파일) 데이터를 파일로 다운받기
9 • SELECT movie_film FROM movieTBL WHERE movie_id=3
10    INTO DUMPFILE 'C:/sql/movies/Mohican_out.mp4';

```

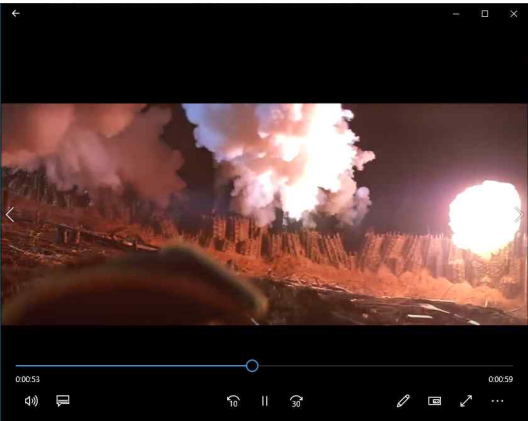
movies

파일 | 홈 | 공유 | 보기

로컬 디스크 (C:) > sql > movies

이름	수정한 날짜	유형	크기
Movies.zip	2018-10-21 ...	ALZip ZIP File	43,035KB
Mohican.mp4	2017-10-14 ...	MP4 파일	16,994KB
Mohican_out.mp4	2020-11-05 ...	MP4 파일	16,994KB
Schindler.mp4	2017-12-24 ...	MP4 파일	13,953KB
Shawshank.mp4	2017-12-24 ...	MP4 파일	17,005KB
Mohican.txt	2018-09-06 ...	텍스트 문서	6KB
Schindler.txt	2018-09-05 ...	텍스트 문서	9KB
Schindler_out.txt	2020-11-05 ...	텍스트 문서	9KB
Shawshank.txt	2018-09-06 ...	텍스트 문서	5KB

9개 항목



Schindler_out.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

노아보다 더 훌륭한 인물이라고 평한 자도 있었다고 하는데, 물론 단순 비교야 곤란하겠지만, 어쨌든, 성경의 노아 못지않게, 그의 이 자랑스러운 이름, "쉴들러" 역시 앞으로 역사에 영원히 남을 것 이다.

가죽은 가죽을 남기고 사람은 역시 이름을 남긴다는데.....

우리는 과연 무엇을 남길까 하는 생각도 이 영화를 보면서 세삼 들기도 한다.

이 영화는 기획을 하고 막상 제작에 착수 하는 데만 약10년이 걸렸다고 한다.

"Jaws"(1975)나 "Indiana Jones"(1981), "ET"(1982)등으로 당시 할리우드에서 이미 거를 영화인으로 자리를 잡은

"Steven Spielberg"(1946, 미국 오하이오/위의 사진 왼쪽사람)로서는 아마 생애 최고의 작품을 만든 것이 아닌가 싶고, 처음에는 실제로 "홀로코스트"를 경험한바있는 "로만 폴란스키"가 감독직을 제의 받았다고 하는데, 만일에 그가 만들었다면 이 영화는 또 어떻게 달라졌을까 하는 호기심이 들기도 한다.

(이 영화에 관여 하지 않은 "폴란스키"는 이후, 21세기에 들어서 2002년도에 "The Pianist" 라는 작품을 만들었다)

여하튼 동료제작자들에게 이 영화는 반드시 흑백으로 제작을 해야만 한다고 고집하여 그들을 설득한 "스티븐 스피버그"의 기획은 다시 한번 성공을 하였고, 인류의 선과 악을 동시에 기록한 하나의 귀중한 역사적인 증거물로 길이 남게 되었다.

"Liam Neeson"(1952, 북아일랜드) 는

Ln 1, Col 1 100% Windows (CRLF) UTF-8(BOM)