MySQL FAQ

version 1.0

- 차례 -

작성자 : 허정수(wertyu@nownuri.net)

작성일 : 2001 년 5 월 23 일

0.0) About This Document

본 문서는 필자가 저술한 책 'MySQL Advanced Class(출판사:베스트 북, ISBN : 89-8397-071-5, 출판일 : 2001 년 4 월 10 일) 의 부록 '허의원의 진료실' 부분에서 Section 1,7 의 내용을 편집한 문서입니다.
필자는 현재까지(2001 년 5 월) MySQL 에 대하여 두 권의 책을 집필하였습니다. 필자가 집필한 책에 관심 있는 분은 http://www.nnr.or.kr/book/diff.php 을 참고하시기 바랍니다.
마음 같아서는 집필한 책을 모두 인터넷 상에 공개하고 싶지만, 출판사로부터 받은 원고비가 있고, 또 돈을 주고 산 독자도 있어서, 그렇지 못하고 있습니다. 먼 훗날 책이 더 이상 안 팔릴 때쯤 차례로 공개하겠습니다.(그때쯤이면 제 책이 더 이상 활용될 수 없을지도 모르겠네요^^)
이 문서는 http://www.nnr.or.kr/ 에서 구할 수 있습니다. 새로운 버전의 문서가 나오면, 항상 위의 웹 사이트에 게시할 테니 오래된 문서를 보는 분들은 위의 웹 사이트를 방문해 보시기 바랍니다.

0.1) Copyright

1 Connection Error

처음으로 Connection Error 에 대해서 알아 보겠습니다. 다음과 같이 두 내용에 대해 설명드리 겠습니다.

```
i : Access Denied
ii: Can't connect to local MySQL server through socket '/tmp/mysql.sock'
```

위의 i, ii 문제는 MySQL 을 처음 사용해 본 초보자들이 가장 많이 겪는 문제입니다. 이 문서를 통하여 위의 문제를 해결하는 방법을 확실히 배우도록 합시다.

여기서 i 문제는 MySQL 의 데몬이 확실히 실행되고 있다는 것은 알 수 있으므로, 그나마 해결하기 쉽습니다. 하지만, ii 문제는 MySQL 데몬이 실행 중인지 아닌지 먼저 알아야 하므로, 약간 더 어려울 수 있습니다.

1.1) Access denied

'Access denied' 에러는 ii 의 경우와 달리 현재 MySQL 데몬이 실행 중이라는 것입니다. 따라서 ii 경우보다는 문제를 해결하기가 쉽습니다.
이 에러 메시지는 다음과 같은 형식으로 출력됩니다.

[wertyu@inos ~]$ mysql -u root student
ERROR 1045: Access denied for user: 'root@localhost' (Using password: NO)

이 에러 메시지가 의미하는 것은 입력한 사용자에게 MySQL 을 사용할 권한이 없다는 것을 의미합니다. 암호를 잘못 지정하였거나, 사용자 명을 잘못 지정한 경우 또는 해당 데이터베이스나 테이블에 접근할 수 있는 권한이 없을 때 발생하는 에러입니다.
MySQL 을 처음 설치한 후 아무런 변경도 하지 않는 경우에는 root 사용자의 암호가 설정되어 있지 않습니다. 따라서 MySQL 을 처음 설치한 경우에는 다음과 같은 명령으로 MySQL 서버에 연결을 할 수 있습니다.

[wertyu@inos ~]$ mysql -u root mysql

여기서 주의할 점은 mysql 은 -u 옵션으로 사용자 명을 주지 않는 경우 시스템 계정 사용자의 이름을 이용한다는 것입니다. 현재 필자는 wertyu 라는 사용자로 작업을 하고 있는데 -u 옵션을 지정하지 않는 이상 MySQL 에서도 wertyu 사용자로 로긴을 하려고 합니다. 그런데 MySQL 에는 초기에 wertyu 사용자가 존재하지 않으므로 다음과 같이 Access denied 에러가 발생하게 됩니다.

[wertyu@inos ~]$ mysql mysql
ERROR 1045: Access denied for user: 'wertyu@localhost' (Using password: NO)

따라서 -u 옵션으로 사용자 명을 지정해 주어야 합니다. 하지만 주의해야 할 점은 MS Windows 에서 mysql 을 사용하는 경우 dos 용 프로그램을 실행하므로 사용자 계정이라는 것이 없고 따라서 anonymous 사용자로 로긴을 시도하게 됩니다. anonymous 사용자란 mysql 데이터베이스에서 User 필드에 값이 없는 사용자를 의미합니다. 이 anonymous 사용자는 MySQL 에 테스트 용으로 접근을 할 수 있는 사용자이지만 아무런 권한이 없는 사용자입니다. 따라서 MS Windows 사용자는 꼭 -u 옵션으로 사용자 명을 적어주어야 합니다. anonymous 로 mysql 을 사용하는 경우 MySQL 에 접속은 할 수 있지만, 아무런 권한이 없는 anonymous 사용자이기 때문에 데이터베이스를 사용하려고 use mysql 과 같은 SQL 문을 실행하면 다음과 같이 Access denied 에러가 납니다.

mysql> use mysql;
ERROR 1044: Access denied for user: '@localhost' to database 'mysql'

따라서 처음 MySQL 을 사용하는 독자들은 로긴은 되었는데 데이터베이스를 사용할 때 Access denied 에러가 나므로 헷갈려하는 경우가 종종 발생합니다.

만약 mysql 을 사용하면서 다음과 같은 에러가 발생한다면 사용자에게 암호가 걸려 있는 것입니다.

[wertyu@inos ~]$ mysql -uroot mysql
ERROR 1045: Access denied for user: 'root@localhost' (Using password: NO)

이 경우 root 사용자는 MySQL 의 사용자인데 mysql 로 로긴을 하려고 했더니, (Using password:No) 를 출력하면서 연결할 수 없다는 에러 메시지를 출력하였습니다. 즉, 이 사용자에게는 암호가 있는데 암호를 입력하지 않았으므로 로긴을 못 한다는 이야기입니다. 그런데 만약 -u 로 사용자를 지정할 때 MySQL 에 없는 사용자를 지정하면 위와 같은 에러가 발생하게 됩니다. 만약 위와 같은 에러가 발생할 때는 MySQL 에 있는 사용자인지 검사를 해보고 MySQL 에 있는 사용자라면 다음과 같이 암호를 입력하면 되고, MySQL 에 없는 사용자라면 사용자를 추가하셔야 합니다.

[wertyu@inos ~]$ mysql -uroot -p mysql
Enter password:

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20 to server version: 3.23.32

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>
```

또한 MySQL 을 처음 사용하는 사람들이 많이 실수하는 것 중의 하나가 사용자를 추가할 때 INSERT 나 UPDATE 등으로 사용자를 추가하면서 FLUSH PRIVILEGES; 를 실행하지 않는 경우입니다. MySQL 은 처음 실행을 할 때 MySQL 의 권한 테이블인 mysql 데이터베이스에 있는 테이블의 내용을 메모리로 읽어 들입니다. 이러한 이유는 사용자 인증 검사 속도를 빠르게 하기 위해서 입니다. 그런데, INSERT 나 UPDATE 등으로 사용자 정보를 변경한 경우 실제 테이블에 내용은 변경되지만 메모리에 있는 내용은 변경이 되지 않습니다. 따라서 초보자들은 'INSERT 로 사용자를 추가했는데, 로긴이 안 된다'라고 많이 합니다. 이렇게 INSERT 나 UPDATE 로 사용자 정보를 변경한 경우에는 꼭 FLUSH PRIVILEGES 로 MySQL 에게 '사용자 정보가 변경되었으니 권한 테이블을 다시 메모리로 읽어 들여라' 라고 알려주어야 합니다. GRANT 문으로 사용자를 추가하면 FLUSH PRIVILEGES 를 할 필요가 없기 때문에 실수를 미연에 방지할 수 있고, 또한 각 사용자에 대해 세세한 권한을 쉽게 줄 수 있으므로 사용자를 추가할 때는 GRANT 문을 이용하시기 바랍니다.(GRANT 에 대해서는 부록 1 참고)

요즘은 PHP 를 많이 사용하는데 PHP 에서 PHP 에서는 다음과 같이 mysql_connect() 함수로 MySQL 에 로긴을 하게 됩니다.

```
$conn = mysql_connect("localhost", "phpuser", "secret") ;
```

그런데 만약 이 함수에서 Access denied 에러가 발생하는 경우 문제를 쉽게 해결하는 방법은 직접 mysql 프로그램을 이용해서 다음과 같이 사용자 명에 phpuser, 암호에 secret 를 입력하여 로긴이 가능한지 살펴보는 것입니다.

```
$ mysql -u phpuser -psecret
```

만약 로긴이 가능하다면 PHP 프로그램이 잘못된 것이고, 로긴을 할 수 없다면, mysql_connect()에서 지정한 사용자명과 암호가 잘못된 것이므로 올바른 사용자명과 암호를 지정해 주어야 합니다.

MySQL 을 설치하면 mysqlaccess 라는 유틸리티가 기본적으로 포함되어 있습니다. 이 프로그램을 이용하여 사용자의 권한과 암호가 걸려 있는지 등등의 정보를 얻을 수 있습니다. 사용법은 다음과 같습니다.

```
mysqlaccess [host [user [db]]] OPTIONS
```

만약 로컬 호스트의 wertyu 에게 student 라는 데이터베이스에 어떤 권한이 있는지 알아보기 위해서는 다음과 같은 명령을 내리면 됩니다.(단, 버그인지 모르겠는데 현재 필자가 사용하는 mysqlaccess 는 시스템의 root 사용자로만 실행을 해야 결과가 나오고 있습니다. 만약 결과가 제대로 나오지 않는 독자는 root 사용자로 mysqlaccess 를 실행시켜보세요)

```
[wertyu@localhost wertyu]# mysqlaccess localhost wertyu student
Could not open outputfile ~/mysqlaccess.log for debugging-info
```

```
mysqlaccess Version 2.06, 20 Dec 2000
By RUG-AIV, by Yves Carlier (Yves.Carlier@rug.ac.be)
Changes by Steve Harvey (sgh@vex.net)
This software comes with ABSOLUTELY NO WARRANTY.

Access-rights
for USER 'wertyu', from HOST 'localhost', to DB 'student'
          +----------------+---+ +----------------+---+
          | Select_priv      | Y | | Shutdown_priv    | N |
          | Insert_priv      | Y | | Process_priv     | N |
          | Update_priv      | Y | | File_priv        | N |
          | Delete_priv      | Y | | Grant_priv       | N |
          | Create_priv      | Y | | References_priv  | Y |
          | Drop_priv        | Y | | Index_priv       | Y |
          | Reload_priv      | N | | Alter_priv       | Y |
          +----------------+---+ +----------------+---+
NOTE:     A password is required for user `wertyu' :-(

The following rules are used:
 db      : 'localhost','student','wertyu','Y','Y','Y','Y','Y','Y','N','Y','Y','Y'
 host    : 'Not processed: host-field is not empty in db-table.'
 user                                                                                :
'localhost','wertyu','4e2eb14c2c15d95e','N','N','N','N','N','N','N','N','N','N','N','N','N'

BUGs can be reported by email to Yves.Carlier@rug.ac.be
```

결과를 보면 처음 'Access-rights' 부분에서 localhost 의 wertyu 사용자에게 student 데이터베이스에 대한 권한들이 출력됩니다. 현재 localhost 의 wertyu 사용자는 student 데이터베이스만 사용할 수 있다는 것을 보여줍니다. 또한 NOTE 부분에서는 wertyu 사용자는 로긴을 하기 위하여 암호가 필요하다는 것을 보여주고 있습니다. 마지막으로 'The following rules are used' 에서는 db 테이블과 host 테이블, user 테이블에서 User 필드가 wertyu 인 레코드들을 출력하게 됩니다.
앞의 결과로서 localhost 의 wertyu 사용자는 암호가 있어야 하고 student 데이터 베이스에만 접근할 수 있다는 것을 보았습니다. 그러면 실제로 mysql 을 사용해서 로긴을 해 볼까요?

```
[wertyu@inos ~]$ mysql -u wertyu -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 33 to server version: 3.23.32

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> use student;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> use mysql
ERROR 1044: Access denied for user: 'wertyu@localhost' to database 'mysql'
```

처음 로긴은 성공으로 하였고 use student 로 student 데이터베이스를 사용하고자 할 때도 성공을 하였으나 use mysql 로 mysql 데이터베이스를 사용하려고 하니 Access denied 에러가 발생하였습니다.

즉, 정리를 하자면 Access denied 에러는 사용자 명을 잘못 지정하였거나 암호를 잘못 지정하

였거나 데이터베이스에 접근할 권한이 없을 때 발생하는 것입니다.

1.2 Can't connect to local MySQL server through socket '/tmp/mysql.sock'

이 문제도 i) 문제와 함께 MySQL 에서 가장 많이 발생하는 에러 중의 하나입니다. i) 문제는 그래도 MySQL 데몬이 실행 중이기 때문에 에러가 발생할 수 있는 원인이 많지가 않아 문제를 해결하기 쉽지만 이 에러는 MySQL 의 데몬이 실행 중이지 않을 수도 있기 때문에 쉽게 해결할 수 없습니다. 하지만 필자가 다음에서 제시하는 방법을 사용하면 초보자라도 쉽게 문제를 해결할 수 있을 것 입니다.
ii) 에러가 발생하는 원인은 크게 다음과 같습니다.

i) MySQL 데몬이 실행되지 않았다.
i) MySQL 데몬에서 생성한 mysql.sock 파일과 mysql 등의 클라이언트 프로그램에서 지정한 mysql.sock 파일의 위치가 서로 다르다
i) MySQL 데몬은 실행 중이나 mysql.sock 파일이 지워졌다.

먼저 정확한 문제 해결을 위해 mysql.sock 파일이 무엇인지 알아 보겠습니다. MySQL 과 클라이언트 프로그램이 서로 연결하는데 사용하는 mysql.sock 파일은 유닉스 소켓 파일이라는 파일입니다.(파일의 이름은 MySQL 을 실행할 때 바꿔줄 수 있는데 기본값은 mysql.sock 파일이니 이 책에서도 mysql.sock 파일이라는 용어를 계속해서 사용하겠습니다.) mysql.sock 파일은 MySQL 서버와 클라이언트 간에 서로 통신을 하는데 사용됩니다. MySQL 과 클라이언트 프로그램(mysql, 혹은 PHP 등의 언어도 모두 클라이언트입니다)이 통신하는 방법에는 유닉스 소켓 파일과 TCP/IP 를 이용하는 방법이 있습니다. 그런데 유닉스 소켓을 사용하는 것이 TCP/IP 를 사용하는 것보다 속도가 빠릅니다. 유닉스 소켓은 속도가 빠르지만 유닉스 소켓은 Local System 에 있는 프로그램 사이에만 서로 연결을 할 수 있습니다. 즉 서버와 클라이언트 모두 한 시스템 안에 있는 경우에만 유닉스 소켓을 사용할 수 있습니다. (MS Windows 에서는 유닉스 소켓을 지원하지 않으므로 TCP/IP 만을 사용하고 이런 문제가 발생하지 않습니다.) 이렇게 되면 로컬에서 MySQL 서버와 클라이언트는 서로 연결할 수 있는 방법이 유닉스 소켓인 mysql.sock 파일과 TCP/IP 를 이용한 방법, 이렇게 두 가지 방법이 있는데 MySQL 의 경우 만약 클라이언트가 로컬 머신에 있는 MySQL 서버에 연결하려고 시도하는 경우 TCP/IP 를 이용하지 않고 무조건 유닉스 소켓 파일을 이용합니다.

하지만 만약 MySQL 서버는 /tmp/mysql.sock 파일을 유닉스 소켓으로 만들었는데, mysql 등의 클라이언트 프로그램은 /usr/local/var/mysql.sock 파일의 위치에서 mysql.sock 파일을 찾으려고 한다면 Can't connect to local MySQL server through socket '/usr/local/var/mysql.sock' 에러가 발생할 수 있습니다. 따라서 이런 경우에는 서로 동일한 위치에서 mysql.sock 파일을 만들고 사용하도록 지정해야 합니다. 또한 MySQL 서버가 실행 중이지 않은 경우도 mysql.sock 파일이 생성되지 않았으므로 이 에러가 발생할 수 있습니다. 그러면 이제 앞에서 설명드린 각 경우 별로 어떻게 문제를 해결할 수 있는지 알아보겠습니다.

i) MySQL 데몬이 실행되지 않은 경우

MySQL 데몬이 실행되지 않은 경우 mysql.sock 파일이 생성되지 않으므로 이런 에러가 발생될 수 있습니다. 일단 이 에러가 발생한 경우에는 MySQL 데몬이 실행 중인지 알아야 합니다. 대부분의 경우 다음과 같은 명령으로 MySQL 데몬이 실행 중인지 알 수 있습니다.

```
[wertyu@inos ~]$ ps auxc | grep mysqld
root      1319  0.5  0.6  1712   852 pts/0    S     12:40   0:00 safe_mysqld
root      1343  1.3  3.3 25436 4264 pts/0    S     12:40   0:00 mysqld
root      1345  0.0  3.3 25436 4264 pts/0    S     12:40   0:00 mysqld
root      1346  0.0  3.3 25436 4264 pts/0    S     12:40   0:00 mysqld
```

```
root        1347   0.0   3.3 25436 4264 pts/0      S      12:40     0:00 mysqld
```

이러한 결과가 출력된다면 현재 MySQL 데몬이 실행 중이라는 것을 알 수 있습니다. 만약 ps 의 옵션으로 auxc 에서 c 옵션을 주지 않은 경우에는 mysqld 프로그램의 경로까지 출력이 되지만 다음과 같이 실제 프로그램 명은 출력이 되지 않으므로 grep 에서 mysqld 를 찾지 못할 수 있습니다. 다음은 ps aux 의 결과인데 프로그램의 경로가 길어서 mysqld 부분이 짤린 결과입니다.

```
[wertyu@inos ~]$ ps aux
..
..
root        1319   0.0   0.6   1712   852 pts/0      S      12:40     0:00 sh /usr/local/mys
root        1343   0.1   3.3 25436 4264 pts/0      S      12:40     0:00 /usr/local/mysql/
root        1345   0.0   3.3 25436 4264 pts/0      S      12:40     0:00 /usr/local/mysql/
root        1346   0.0   3.3 25436 4264 pts/0      S      12:40     0:00 /usr/local/mysql/
root        1347   0.0   3.3 25436 4264 pts/0      S      12:40     0:00 /usr/local/mysql/
wertyu      1357   0.0   0.7   2552   924 pts/0      R      12:42     0:00 ps aux
```

c 옵션은 일단 mysqld 가 실행 중인지 확인을 한 후에 어느 경로에서 실행 중인지 알고 싶을 때는 c 옵션 없이 ps 를 실행하는 것이 좋습니다.

또한 netstat -a 명령으로 다음과 같이 MySQL 데몬이 실행 중인 것을 볼 수 있습니다.

```
[wertyu@inos ~]$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 *:mysql                     *:*                         LISTEN
```

필자의 경우는 *:mysql 이라고 출력이 되었는데 독자들의 환경에 따라서 *:3306 이라고 출력되는 경우도 있으므로 주의하시기 바랍니다.(이는 /etc/services 파일에서 MySQL 의 서비스를 지정하지 않았기 때문입니다.)

MySQL 데몬이 실행 중인 것을 확인하였으면 이제 Can't Connect 에러가 발생하는 이유는 MySQL 데몬이 실행 중이지 않는 이유가 아니라는 것을 알 수 있으므로 이유는 다른 문제에 있으므로 다음을 계속 읽어 주시기 바랍니다. 만약 MySQL 데몬이 실행 중이지 않다면 MySQL 데몬을 실행 시켜 주시고 MySQL 데몬을 실행 시킨 후에도 에러가 계속 발생한다면 다음을 계속해서 읽어 주세요.

i) MySQL 데몬에서 생성한 mysql.sock 파일과 mysql 등의 클라이언트 프로그램에서 지정한 mysql.sock 파일의 위치가 서로 다르다

앞에서 mysql.sock 파일이 무엇인지 어떤 역할을 하는지에 대해서 잠시 설명을 드렸습니다. 중요한 것은 mysql.sock 파일은 로컬의 MySQL 서버와 클라이언트들이 서로 통신을 하는데 사용하는 유닉스 소켓 파일인데 만약 MySQL 서버에서는 mysql.sock 파일을 /tmp/mysql.sock 에 만들었는데 클라이언트 파일은 다른 위치에서 mysql.sock 파일을 찾으려 한다면 에러가 발생한다는 것입니다.
이 문제를 해결하기 위해서는 MySQL 서버에서 mysql.sock 파일을 만든 위치를 찾아야 한다는 것입니다. 그런 후 클라이언트 프로그램을 사용할 때 -S 옵션(혹은 --socket 옵션)으로 직접 mysql.sock 파일의 위치를 지정해 주어야 합니다.
mysql.sock 파일을 찾는 가장 간단한 방법으로는 MySQL 을 설치할 때 기본적으로 포함되는 mysql_config 라는 파일을 이용하는 것입니다. 만약 MySQL 을 설치하고나서 mysql.sock 파일을 위치를 변경하지 않았다면 mysql_config 프로그램으로 mysql.sock 파일의 위치를 찾을 수

있습니다.

```
[wertyu@inos bin]$ mysql_config --socket
/tmp/mysql.sock
```

앞의 결과는 현재 MySQL 데몬은 mysql.sock 파일을 /tmp/mysql.sock 에 만든다는 것을 나타냅니다. 하지만 이 결과는 MySQL 을 사용하면서 충분히 바뀔 수 있는 것이기 때문에 정확한 답을 출력하지 않을 수도 있습니다. 더욱 정확한 답을 찾기 위해서는 netstat -ap 명령을 실행하는 것입니다. 다음은 그 결과입니다.

```
[wertyu@inos bin]$ netstat -ap | grep mysql
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 *:mysql                 *:*                     LISTEN      -
unix  0      [ ACC ]     STREAM     LISTENING     1596   -
/tmp/mysql.sock
```

이 결과를 보면 마지막에 LISTEN 이라는 항목 밑에 /tmp/mysql.sock 이라는 것을 볼 수 있습니다. 이 말은 현재 MySQL 이 /tmp/mysql.sock 파일에서 연결을 기다리고 있다는 말입니다. 이제 /tmp/mysql.sock 파일을 찾으셨습니다. MySQL 이 실행 중이라면 위의 결과가 나오지 않는 경우는 없을 것입니다. mysql.sock 파일이라는 이름을 사용하지 않는 경우에는 grep mysql 로 검색이 되지 않으므로 grep mysql 없이 netstat -ap 의 결과만으로 손수 찾아 보시기 바랍니다.

이렇게 해서 mysql.sock 파일의 위치를 찾은 경우에는 클라이언트 프로그램을 사용할 때 -S 옵션으로 다음과 같이 mysql.sock 파일의 위치를 직접 지정해 주어야 합니다.

```
[wertyu@inos bin]$ mysql -S/tmp/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.32

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>
```

하지만 mysql 같은 크라이언트 프로그램에서는 -S 옵션으로 mysql.sock 파일을 지정하면 되지만 PHP 같은 언어에서는 어떻게 지정할까요? MySQL C API 에서 MySQL 서버에 연결할 때 사용하는 mysql_real_connect() 함수의 가장 마지막 인자는 mysql.sock 파일의 위치를 지정하는 옵션이므로 이 옵션을 이용하면 되고, PHP 에서는 PHP 의 설정 파일인 php.ini 파일에서 다음과 같은 라인을 찾아 수정해 주면 됩니다.

mysql.default_socket   =

여기에 앞에서 찾은 mysql.sock 파일의 위치를 적은 후에 웹 서버를 재 시작하면 문제없이 실행될 것입니다.

다음과 같이 mysqladmin 명령으로도 mysql.sock 파일의 위치를 알아낼 수 있습니다.

```
[wertyu@inos ~]$ mysqladmin variables | grep mysql.sock
| socket                        | /tmp/mysql.sock                                                 |
```

하지만 문제는 mysqladmin 프로그램도 MySQL 클라이언트 프로그램인지라 이 문제를 해결하기 전에는 연결을 할 수 없다는 것입니다.

i) MySQL 데몬은 실행 중이나 mysql.sock 파일이 지워졌다.

만약 MySQL 데몬도 잘 실행 중이고 mysql.sock 파일의 위치도 알아내었은데 계속해서 문제가 발생된다면 mysql.sock 파일이 지워졌을 가능성이 있습니다. 서버를 제대로 설정했다면 이런 일을 발생하지 않지만 가끔 관리자의 부주의 혹은 MySQL 서버에 이상이 생긴 경우 또는 크래커의 장난 등의 이유로 mysql.sock 파일이 지워지는 경우가 있습니다. 이런 경우 일단 ls 명령으로 mysql.sock 파일이 존재하는지 검사를 한 후 정말로 mysql.sock 파일이 없다면 MySQL 데몬의 실행을 중지한 후 재 시작해야 합니다. 주의해야 할 점은 지금 현재의 문제가 확실히 mysql.sock 파일이 지워진 것인지를 알아야 한다는 것입니다. 만약 mysql.sock 파일이 지워져서 생긴 문제가 아니라면 아무리 재시작을 해도 문제가 해결 될리가 없기 때문입니다. 통상적으로 MySQL 데몬은 mysqladmin shutdown 명령으로 실행을 중지시킬 수 있지만(그 외 방법으로 아예 시스템을 재 부팅해도 됩니다) mysqladmin 프로그램도 MySQL 클라이언트 프로그램이기 때문에 이 문제가 발생했을 때는 실행을 시킬 수가 없을 것입니다. 이 경우에는 다음과 같은 명령으로 MySQL 데몬의 실행을 중지시킬 수 있습니다. 다음에서 /path/to/mysql/data/directory 란 MySQL 의 데이터 디렉터리로 이동하라는 것을 의미합니다. MySQL 의 데이터 디렉터리는 $ mysql admin variables | grep datadir 명령으로 확인할 수 있습니다.

```
$ cd /path/to/mysql/data/directory
$ kill `cat hostname.pid`
```

먼저 cd 명령으로 MySQL 의 데이터 디렉터리로 이동합니다.(데이터 디렉터리가 어딘지 모르는 독자는 '부록 1'을 참고하세요) 데이터 디렉터리를 보면 hostname.pid 라는 파일이 존재하는데 이 파일에 현재 MySQL 의 프로세스 아이디가 기록되어 있습니다. 그 후 kill `cat hostname.pid`로 MySQL 데몬의 실행을 중지시킬 수 있습니다. 여기서 `는 '가 아니라 `이니 헷갈리지 않기 바랍니다.

2. PHP 에서의 MySQL 과 관련된 Warning 메시지 해결 방법

PHP 로 프로그래밍을 하다 보면 다음과 같은 Warning 메시지가 출력되는 경우를 많이 볼 수 있습니다.

```
Warning: Supplied argument is not a valid MySQL-Link resource in /hd2/home/wertyu/public_html/error.php3 on line 4

Warning: Supplied argument is not a valid MySQL-Link resource in /hd2/home/wertyu/public_html/error.php3 on line 8

Warning: Supplied argument is not a valid MySQL result resource in /hd2/home/wertyu/public_html/error.php3 on line 10
```

대부분 이런 경우는 프로그램 내에서 에러 검사를 전혀 하지 않았기 때문입니다. 좋은 프로그래머는 항상 함수를 실행한 뒤에 에러 검사를 합니다. 아무리 뛰어난 프로그래머라도 실수를 할 수 있고, 시스템에 이상이 생겨서 에러가 발생할 수 있기 때문이지요. 앞의 Warning 메시지들은 다음의 아주 간단한 프로그램 내에서 발생한 메시지들입니다. 다음의 소스와 앞의 메시지들로부터 어떻게 프로그램을 작성해야 좋은 것인지에 대해서 알아보도록 하겠습니다.

```
1  <?
```

```
 2          $conn = mysql_connect("localhost", "wertyu", "pass") ;
 3
 4          mysql_select_db("nnr", $conn) ;
 5
 6          $query = "SELECT   FROM student" ;
 7
 8          $result = mysql_query( $query, $conn) ;
 9
10          $row = mysql_fetch_array( $result ) ;
11
12          echo $row["student_no"] ;
13
14  ?>
```

먼저 앞의 Warning 메시지를 보면 Supplied argument is not a valid MySQL-Link 라는 말이 나오는데 이 말을 해석해 보면 '입력된 인자는 올바른 MySQL 의 연결이 아니다'라고 해석을 할 수 있습니다. 즉, 이 Warning 메시지는 mysql_connect() 함수에서 연결이 제대로 되지 않았기 때문에 발생하는 Warning 메시지라는 것을 알 수 있습니다. 그러면 다음과 같이 mysql_connect() 함수 부분을 에러 검사를 하도록 수정해 보겠습니다.

```
$conn = mysql_connect("localhost", "wertyu", "pass") ;
if( $conn == 0 )
{
    die( mysql_error() ) ;
}
```

그런 후 다시 실행을 해 보니

```
Warning: MySQL Connection Failed: Access denied for user: 'wertyu@localhost' (Using password: YES) in /hd2/home/wertyu/public_html/error.php3 on line 2
```

와 같은 에러가 출력되면서 프로그램이 종료되었습니다. 즉, 지금까지 문제를 발생시켰던 Waring 메시지의 원인 중 하나는 연결을 못하였기 때문에 발생하였다는 것을 알 수 있었습니다. 이제 mysql_connect() 함수에 사용자 명과 암호를 다시 입력하여 정확히 연결을 할 수 있도록 다음과 같이 수정한 후 계속하여 실행을 해 보았습니다.

```
 1  <?
 2          $conn = mysql_connect("localhost", "wertyu", "new_pass") ;
 3          if( $conn == 0 )
 4          {
 5                  die( mysql_error() ) ;
 6          }
 7
 8          mysql_select_db("nnr", $conn) ;
 9
10          $query = "SELECT   FROM student" ;
11
12          $result = mysql_query( $query, $conn) ;
13
14          $row = mysql_fetch_array( $result ) ;
15
16          echo $row["student_no"] ;
17
18  ?>
```

그랬더니 이번에는 다음과 같은 에러가 발생하는군요.

Warning: Supplied argument is not a valid MySQL result resource in /hd2/home/wertyu/public_html/error.php3 on line 14

이번 에러를 해석해 보면, '입력된 인자는 옳바른 MySQL 의 결과가 아니다'라고 해석할 수 있습니다. 즉, 이 말이 의미하는 것은 mysql_query() 함수를 실행하여 쿼리의 결과를 리턴하는데 앞의 예제에서 mysql_query() 함수에서 이상이 발생하였다는 것을 의미하는 것입니다. 따라서 이번에도 mysql_query() 함수에서 에러 검사를 하는 코드를 다음과 같이 입력하시기 바랍니다.

```
$result = mysql_query( $query, $conn) ;
if( $result == 0 )
{
    die( mysql_error() ) ;
}
```

그런 후 다시 실행을 하니 다음과 같은 Warning 메시지가 출력되었습니다.

You have an error in your SQL syntax near 'FROM student' at line 1

아... mysql_query()에서 사용한 $query = "SELECT FROM student" 에서 SELECT 의 문법이 잘못되었군요. 이렇게 에러 검사를 하면 조금 더 쉽게 에러를 검사할 수 있습니다. 앞의 예에서는 쿼리가 너무나 간단하여 쉽게 눈으로도 에러를 검사할 수 있지마나 몇 줄에 걸쳐있는 쿼리들은 에러를 잡기가 쉽지 않습니다. Syntax 에러가 발생하는 경우, echo $query 와 같이 쿼리를 직접 출력해서 확인하는 것도 좋은 방법입니다. 그런데, 가끔 쿼리를 mysql_query() 함수안에서 정의하는 분들이 있는데, 이렇게 되면, 쿼리를 echo 로 출력할 수 없습니다. 따라서 좋은 습관은 쿼리를 mysql_query() 함수 밖에서 정의하고, 함수의 인자로 넘기는 방법입니다. 이렇게 해야 다음에 echo $query 로 쿼리가 어떻게 되었는지 눈으로 확인하기 쉽기 때문에, Syntax 에러를 쉽게 찾을 수 있습니다.

자, 이제 쿼리를 수정도 하고 해서 다음과 같은 PHP 코드가 만들어졌습니다.

```
1  <?
2          $conn = mysql_connect("localhost", "wertyu", "new_pass") ;
3          if( $conn == 0 )
4          {
5                  die( mysql_error() ) ;
6          }
7
8          mysql_select_db("nnr", $conn) ;
9
10         $query = "SELECT   FROM student" ;
11
12         $result = mysql_query( $query, $conn) ;
13         if( $result == 0 )
14         {
15                 die( mysql_error() ) ;
16         }
17
18         $row = mysql_fetch_array( $result ) ;
19
20         echo $row["student_no"] ;
```

```
21
22  ?>
```

이제 제대로 실행되는지 다시 한번 실행해 봅시다. 이번에는 제대로 실행될 줄 알았는데 이번에는 다음과 같은 에러가 출력되는군요.

```
No Database Selected
```

독자의 환경 설정에 따라 다른 메시지로 출력되는 경우가 있습니다. 이번에는 대체 어디가 잘못된 것일까요? 잘 한번 맞춰보세요. 의심이 가는 곳이라고는 딱 한군데 mysql_select_db() 부분입니다. 그 부분에서만 에러 검사를 하지 않았기 때문입니다. 이제 mysql_select_db()도 다음과 같이 에러 검사를 하도록 수정하시기 바랍니다.

```
if( ! mysql_select_db("nnr", $conn) )
{
    die( mysql_error() );
}
```

위와 같이 mysql_select_db() 함수에서도 에러 검사를 하고 실행을 해보니 다음과 같은 Warning 메시지가 출력되었습니다.

```
Access denied for user: 'wertyu@localhost' to database 'nnr'
```

즉, 현재 사용자는 로긴을 할 수는 있지만 nnr 이라는 데이터 베이스에 접근을 할 수 없었어서 생기는 문제였습니다. 이제 wertyu 사용자에게 적절한 권한을 주고 다시 실행을 시키면 아무 문제가 없겠습니다. 최종 완성된 코드는 다음과 같습니다.

```
 1  <?
 2          $conn = mysql_connect("localhost", "wertyu", "new_pass") ;
 3          if( $conn == 0 )
 4          {
 5                  die( mysql_error() ) ;
 6          }
 7
 8          if( ! mysql_select_db("nnr", $conn) )
 9          {
10                  die( mysql_error() );
11          }
12
13          $query = "SELECT *    FROM student" ;
14
15          $result = mysql_query( $query, $conn) ;
16          if( $result == 0 )
17          {
18                  die( mysql_error() ) ;
19          }
20
21          $row = mysql_fetch_array( $result ) ;
22
23          echo $row["student_no"] ;
24
25  ?>
```

다시 한번 말씀 드리지만 훌륭한 프로그래머는 에러 검사를 정확히 하여 프로그램을 오류가 없도록 만듭니다. 또한 함수를 사용할 때도 메뉴얼 등을 읽어 가면서 이 함수의 return 값이

무엇인지, 함수 호출이 실패되었을 경우에 리턴되는 값과 성공하였을 때 리턴되는 값을 비교하여 항상 에러 검사를 합니다.

독자 여러분들도 에러 검사를 확실히 하시기 바랍니다.

마지막으로 다음의 에러를 보겠습니다.

Warning: Unable to jump to row 0 on MySQL result index 2 in /home/kimsh/public_html/sarang/res.php on line 6

Warning: Unable to jump to row 0 on MySQL result index 2 in /home/kimsh/public_html/sarang/res.php on line 7

Warning: Unable to jump to row 0 on MySQL result index 2 in /home/kimsh/public_html/sarang/res.php on line 8

```
5 : $dbresult = mysql_query("select * from res where res='result'",$dbconn);
6 : $rgood = mysql_result($dbresult,0,"good");
7 : $rnormal = mysql_result($dbresult,0"normal");
8 : $rbad = mysql_result($dbresult,0,"bad");
```

이 에러는 mysql_query()에서 함수를 실행한 후, mysql_result()로 레코드를 하나씩 가져오는데, 쿼리의 결과로 가져온 레코드가 하나도 없는데, 레코드를 가져오려고 하기 때문에 발생하는 에러입니다. 이런 경우는 5 번 라인과 6 번 라인 사이에, mysql_num_rows() 함수로 몇 개의 레코드가 있는지 확인을 한 후, 1 개 이상의 레코드만 있는 경우, mysql_result() 함수를 사용하도록 고쳐야 합니다. 그리고, 위에서도 쿼리를 mysql_query() 함수 안에서 적었는데, 함수 밖에서 query 를 정의하고, echo $query 로 어떤 쿼리가 실행되었는지 눈으로 확인한 후에, 왜 결과가 없었는지 보는 것도 좋은 습관입니다.