jinyPHP 2권 - 005. 테이블

**05. 테이블**

====================

데이터베이스 시스템은 다수의 데이터베이스 이름 그룹과 그룹속에 속하는 다수의 테이블 구조의 파일들을 담고 있습니다. 또한, SQL 명령을 통하여 관련 데이터를 조작하고 자료를 검색합니다.

테이블은 실제적인 데이터를 담고 있는 파일과 같습니다. 우리가 흔히 사용하는 엑셀과 같은 간단한 개인형 데이터베이스 응용 프로그램의 경우 .xls 파일을 생성하여 그 안에 데이터를 담고 있습니다. 그리고 이러한 .xls 엑셀 파일은 용도에 맞게끔 그룹화된 폴더로 관리하고 있을 것입니다.

데이터베이스 이름은 여러 개의 테이블 데이터를 관리하기 위한 그룹과 같습니다. 데이터베이스 명의 생성은 테이블을 보관할 수 있는 물리적인 폴더 형태의 디렉토리를 생성합니다. 데이터베이스에는 실제적인 데이터를 가지고 있는 다수의 테이블이 존재합니다. 테이블 또한 테이블 생성과 동시에 관련 테이블 파일이 물리적으로 생성이 됩니다.

**05.1 테이블**

====================

보통 테이블은 초기 DB의 구조를 설계할 때 한번 생성을 하는 경우가 많이 있습니다. 이런경우, 전용 GUI 기반의 DB관리툴을 이용하는 경우가 많이 있습니다. DB관리툴로 오라클에서 배포하는 워크벤치, 웹기반의 관리툴인 phpmyadmin 와 같은 다양한 무료 유틸리티 등이 있습니다.

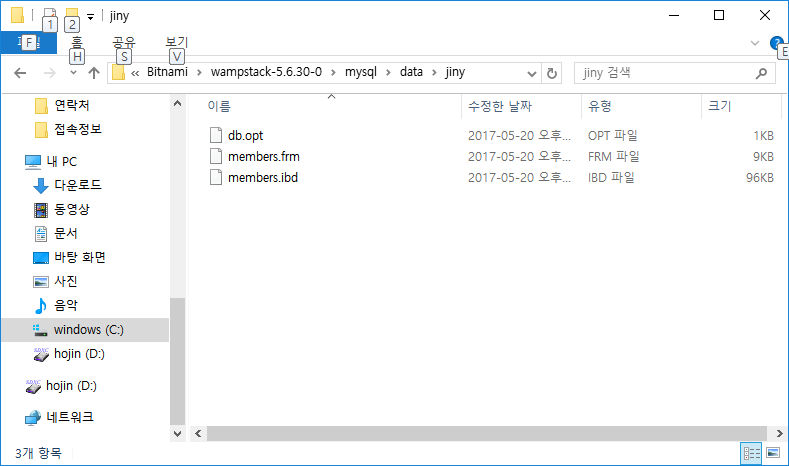
또한, 데이터베이스의 테이블은 SQL 명령을 통하여 생성을 할 수도 있습니다. 테이블을 생성하는 SQL 명령문 으로는 CREATE TABLE 을 사용하면 됩니다. 작성 쿼리 문법은 다음과 같습니다.

**쿼리문법)**

|  |
| --- |
| CREATE TABLE 테이블명 (  컬럼명 선언 ...  컬럼명 데이터타입 컬럼속성,  컬럼명 데이터타입 컬럼속성,  컬럼명 데이터타입 컬럼속성  ); |

데이블 생성 명령과 테이블명, 소괄호 안에 포함될 컬럼의 정보들을 나열하면 됩니다. 다수의 컬럼들은 콤마(,) 기호로 구분되어 정의합니다. 위와 같이 형식으로 SQL 문법을 작성하면 테이블을 생성합니다.

테이블을 생성후에 실제적으로 MYSQL 데이터가 저장되어 있는 경로를 확인해보 면은 데이터 베이스명의 폴더 안에 생성한 테이블 파일들을 확인해 볼 수 있습니다.



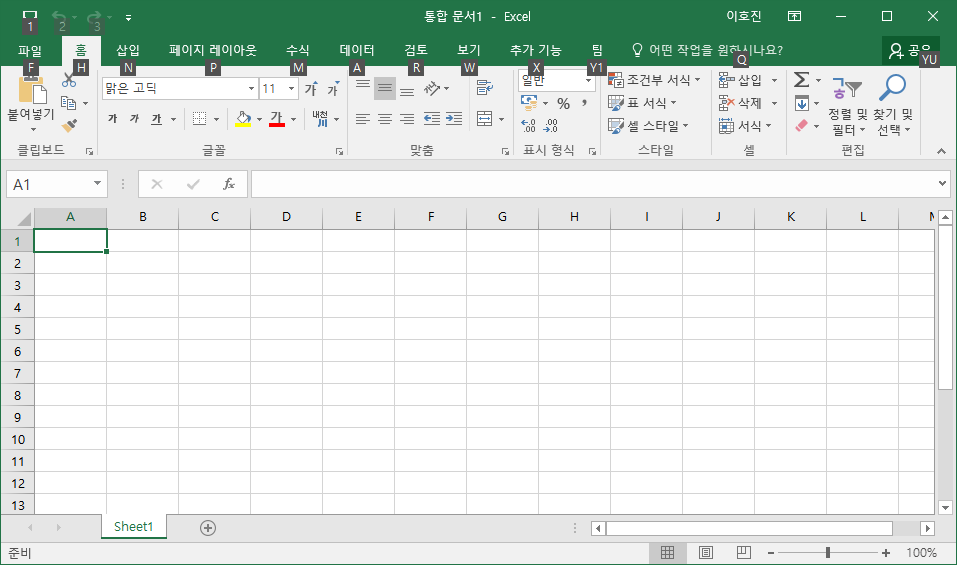
위의 파일구조는 데이터베이스명 및 테이블의 개념을 설명하기 위해서 화면을 캡처한 것입니다. MYSQL 및 SQL 명령으로 접근해야 됩니다. 파일을 직접 조작을 해서는 안됩니다.

**05.1.1 컬럼과 로우**

====================

데이블은 실제적인 데이터를 셀 형태로 저장하고 보관합니다. 데이블은 컬럼이라는 기준으로 각각의 데이터 형식을 구분하고, 컬럼의 맞는 데이터를 열(rows)로 추가하여 데이터를 가지고 있습니다.

우리가 일반적으로 많이 사용하는 엑셀의 경우 다수의 사각박스 형태의 그리드 모양을 보실 수 있습니다. 하나 하나의 박스가 그리드 입니다. 여기에는 데이터값을 입력할 수 있는 저장 영역 입니다.



위처럼 위의 A, B, C 형태로 이름이 붙어 있는 부분이 컬럼 영역이라고 생각하시면 됩니다. 데이터 베이스에서는 컬럼명의 이름을 미리 다 지정을 해주어야 하지만 엑셀의 경우 미리 알파벳으로 지정해 놓고 있는 것입니다. 컬럼은 다른말로 필드(field)라는 표현으로도 불리웁니다.

다음 왼쪽에 숫자로 되어 있는 칸이 있습니다. 열(Rows) 이라고 합니다. 각각의 컬럼에 맞게 데이터가 순차적으로 입력되는 영역입니다. 데이터가 컬럼 양식에 맞게 열이 추가되면서 하나씩 증가 됩니다. 열(Rows)은 다른 말로 레코드(Record)라고도 불리웁니다.

테이블은 엑셀과 같이 컬럼과 열의 실제적인 데이터를 가지는 물리적 파일입니다.

**05.1.2 칼럼 이름**

====================

테이블에는 최소 한 개 이상의 컬럼을 선언하여야 합니다. 칼럼명은 중복하여 이름을 저장할 수 없습니다. 한개의 테이블 안에서의 컬럼은 유일한 이름을 가지게 됩니다. 중복하여 이름을 지정할 경우 시스템은 오류를 발생합니다.

**05.1.3 자료형**

====================

테이블의 칼럼은 데이터를 담을 수 있는 변수와 같습니다. 컬럼의 이름을 지정하는 것과 동시에 데이터의 자료형 타입도 같이 지정을 해주어야 합니다. 데이터 타입을 지정하는 것은 데이터 유형별로 자료구조가 틀리며, 저장 공간을 효율적으로 사용을 하기 위해서 입니다.

대표적으로 int, varchar, text 등 MYSQL 또한 다양한 데이터 자료형이 있습니다.

**05.1.4 속성**

====================

테이블은 컬럼의 이름과 자료형 타입 이외에 추가로 속성을 지정할 수 있습니다. 속성을 지정하게 되면 속성 특성에 따라서 데이터의 잘못된 오류입력과 처리를 방지하고, 데이터베이스의 동작 성능을 향상시킬 수 있습니다.

MYSQL은 5가지 타입의 속성을 지원합니다.

* NOT NULL - NULL값의 허용 여부를 지정합니다. not null을 체크했을 때는 반드시 데이터 값을 포함되어야 합니다.
* DEFAULT 값 - 기본적으로 설정되는 초기값을 지정할 수 있습니다.
* UNSIGNED – 자료형이 숫자 타입일 경우 0 을 포함한 양의 정수만 저장 가능합니다.
* AUTO INCREMENT – 값이 1씩 자동으로 증가되는 속성입니다.
* PRIMARY KEY – 테이블의 열(rows)을 구분할 수 있는 유일한 값을 가집니다.

**05.2 테이블생성**

====================

SQL 명령문을 통하여 실제적인 테이블을 생성해 봅니다. 테이블 생성은 테이블명과 ( ) 안에 컬럼 정보를 같이 선언하여 주면 됩니다.

**CREATE TABLE 테이블명 (**

**컬럼정보…**

**)**

테이블 생성시 들어가는 컬럼은 **컬럼명, 데이터타입, 컬럼속성** 3가지의 값을 한줄로 선언을 하면 됩니다. 또한 여러 개의 컬럼 설정은 콤마(,)러 구분합니다.

**컬럼명 데이터타입 컬럼속성,**

컬럼 속성은 생략이 가능합니다. 하지만, 하나의 테이블에는 반드시 Primary키 속성을 가지고 있는 컬럼을 포함하고 있어야 합니다. 주로 프라이머리 키로 컬럼명을 "id"로 사용하는 경우가 많습니다. 프라이머리 키는 각 레코드데이터는 중복되지 않는 유일한 키 값을 가지고 있습니다.

**05.2.1 콘솔실습**

====================

다음 예제 쿼리는 회원목록을 담고 있는 테이블입니다. SQL 쿼리를 통하여 테이블을 생성해 봅니다.

|  |
| --- |
| CREATE TABLE `members` (  `Id` int(11) NOT NULL AUTO\_INCREMENT,  LastName varchar(255),  FirstName varchar(255),  Address varchar(255),  City varchar(255),  Country varchar(255),  PRIMARY KEY (`Id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; |

예제 쿼리를 콘솔에 입력하여 테이블을 생성해 보도록 실습해 봅니다.

콘솔 출력화면)

|  |
| --- |
| mysql> CREATE TABLE `members` (  -> `Id` int(11) NOT NULL AUTO\_INCREMENT,  -> LastName varchar(255),  -> FirstName varchar(255),  -> Address varchar(255),  -> City varchar(255),  -> Country varchar(255),  -> PRIMARY KEY (`Id`)  -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  Query OK, 0 rows affected (0.04 sec) |

테이블생성시 추가로 DB 엔진과 언어셋을 같이 설정하여 생성할 수도 있습니다. 이에 대해서는 다음에 다시 설명하도록 하겠습니다.

**05.2.2 PHP 실습**

====================

PHP를 통하여 소스상에서 쿼리 전송을 통한 테이블을 생성해 보도록 하겠습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다. 테이블을 생성하는 메서드를 추가합니다.

|  |
| --- |
| // 테이블 생성  public function createTable($queryString){  $this->msgEcho($queryString);  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 테이블 생성!");  // 객체 반환, 매서드체인  return $this;  } else {  $this->msgEcho("Error] ".$queryString);  }  } |

**예제파일) sql-11.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  $queryString = "  CREATE TABLE `members123` (  `Id` int(11) NOT NULL AUTO\_INCREMENT,  LastName varchar(255),  FirstName varchar(255),  Address varchar(255),  City varchar(255),  Country varchar(255),  PRIMARY KEY (`Id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;";  // 테이블을 생성합니다.  $db->createTable($queryString);  ?> |

출력화면)

mysql connected!

CREATE TABLE `members123` ( `Id` int(11) NOT NULL AUTO\_INCREMENT, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255), Country varchar(255), PRIMARY KEY (`Id`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

쿼리성공] CREATE TABLE `members123` ( `Id` int(11) NOT NULL AUTO\_INCREMENT, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255), Country varchar(255), PRIMARY KEY (`Id`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

데이터베이스 삭제!

PHP를 통해서 테이블 생성 쿼리를 전송합니다. 실제적으로 데이터베이스에 테이블이 생성이 되었습니다.

**05.3 테이블확인**

====================

한 개의 데이터베이스 이름 그룹안에는 다수의 테이블을 포함하고 있습니다. 데이터베이스 안에 있는 테이블의 목록을 SQL 명령을 통해서 확인을 할 수 있습니다.

**05.3.1 테이블 목록**

====================

현재 선택된 데이터베이스 이름 그룹안에 들어 있는 모든 테이블 명을 출력합니다. 실행 명령은 SHOW TABLES; 입니다.

**쿼리문법)**

|  |
| --- |
| SHOW TABLES; |

콘솔 실습)

|  |
| --- |
| mysql> show tables;  +----------------+  | Tables\_in\_jiny |  +----------------+  | members |  +----------------+  1 row in set (0.00 sec) |

위와 같이 콘솔로 확인해보면 생성된 테이블의 목록을 확인할 수 있습니다.

**05.3.2 목록 출력**

====================

PHP를 통하여 데이터베이스 이름 그룹안에 있는 테이블의 목록을 출력해 보도록 하겠습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| // 데이터베이스 목록을 읽어 옵니다  public function showTables()  {  $queryString = "show tables";  $this->msgEcho($queryString);  if ($result = mysqli\_query($this->dbcon, $queryString)) {  $rowss = "";  $row\_cnt = mysqli\_num\_rows($result);  for ($i=0;$i<$row\_cnt;$i++) {  $rowss[$i] = mysqli\_fetch\_object($result);  }  $result->free();  return $rowss;  }  } |

**예제파일) sql-12.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 현재 데이터베이스 이름을 확인합니다.  $dbname = $db->currentDatabase();  echo "현재 데이터베이스 = ". $dbname . "<br>";  $key = "Tables\_in\_".$dbname;  if($rowss = $db->showTables()){  echo "tables Count = ". count($rowss) . "<br>";  for($i=0;$i<count($rowss);$i++){  echo $i."=";  echo $rowss[$i]->$key;  echo "<br>";  print\_r($rowss[$i]);  echo "<br>";  }  }  ?> |

출력화면)

mysql connected!

SELECT DATABASE()

현재 데이터베이스 = jiny

show tables

tables Count = 5

0=members

stdClass Object ( [Tables\_in\_jiny] => members )

1=members1

stdClass Object ( [Tables\_in\_jiny] => members1 )

2=members5

stdClass Object ( [Tables\_in\_jiny] => members5 )

3=orders1

stdClass Object ( [Tables\_in\_jiny] => orders1 )

4=products

stdClass Object ( [Tables\_in\_jiny] => products )

**05.3.3 테이블 상세정보**

====================

SHOW TABLES 명령은 데이터베이스의 테이블의 이름목록을 출력합니다. 각각의 테이블의 상세 적인 정보를 알고 싶을 경우에는 뒤에 STATUS 명령을 같이 사용합니다.

**쿼리문법)**

|  |
| --- |
| SHOW TABLE STATUS; |

SHOW TABLE STATUS; 쿼리명령은 테이블 목록과 더불어 테이블의 상세 정보도 같이 출력을 합니다.

콘솔 출력화면)

|  |
| --- |
| mysql> show table status;  +-------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+---------------------+-----------------+----------+--------------------+---------+  | Name | Engine | Version | Row\_format | Rows | Avg\_row\_length | Data\_length | Max\_data\_length | Index\_length | Data\_free | Auto\_increment | Create\_time | Update\_time | Check\_time | Collation | Checksum | Create\_options | Comment |  +-------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+---------------------+-----------------+----------+--------------------+---------+  | board | InnoDB | 10 | Compact | 2 | 8192 | 16384 | 0 | 0 | 0 | 7 | 2017-06-11 15:07:15 | NULL | NULL | utf8\_general\_ci | NULL | | |  | mem\_view | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | VIEW |  | members | MyISAM | 10 | Dynamic | 7 | 45 | 316 | 281474976710655 | 2048 | 0 | 101 | 2017-06-07 17:26:47 | 2017-06-07 17:27:46 | 2017-06-11 14:47:08 | utf8\_general\_ci | NULL | | |  | members1 | MyISAM | 10 | Dynamic | 19 | 23 | 448 | 281474976710655 | 2048 | 0 | 20 | 2017-06-01 14:53:34 | 2017-06-06 18:03:45 | NULL | utf8\_general\_ci | NULL | row\_format=DYNAMIC | |  | members3 | MyISAM | 10 | Dynamic | 0 | 0 | 0 | 281474976710655 | 1024 | 0 | 8 | 2017-06-06 18:55:10 | 2017-06-06 18:55:10 | NULL | utf8\_general\_ci | NULL | row\_format=DYNAMIC | |  | members\_1 | MyISAM | 10 | Dynamic | 6 | 48 | 288 | 281474976710655 | 2048 | 0 | 8 | 2017-06-01 23:55:05 | 2017-06-02 00:20:22 | NULL | utf8\_general\_ci | NULL | | |  | members\_all | InnoDB | 10 | Compact | 0 | 0 | 16384 | 0 | 0 | 0 | NULL | 2017-06-02 16:59:01 | NULL | NULL | utf8\_general\_ci | NULL | | |  | orders | InnoDB | 10 | Compact | 10 | 1638 | 16384 | 0 | 0 | 0 | 11 | 2017-05-22 15:33:25 | NULL | NULL | utf8\_general\_ci | NULL | | |  | products | InnoDB | 10 | Compact | 2 | 8192 | 16384 | 0 | 0 | 0 | 4 | 2017-05-21 12:50:22 | NULL | NULL | utf8\_general\_ci | NULL | | |  +-------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+---------------------+-----------------+----------+--------------------+---------+  9 rows in set (0.06 sec) |

목록과 출력되는 테이블의 상세 정보는 다음과 같습니다.

* Name
* Engine
* Version
* Row\_format
* Rows
* Avg\_row\_length
* Data\_length
* Max\_data\_length
* Index\_length
* Data\_free
* Auto\_increment
* Create\_time
* Update\_time
* Check\_time
* Collation
* Checksum
* Create\_options
* Comment

또는 쉘 명령창에서 **mysqlshow -- status DB명** 으로 확인을 할 수도 있습니다.

|  |
| --- |
| C:\Bitnami\wampstack-5.6.30-0\mysql\bin>mysqlshow -u root -p --status jiny  Enter password: \*\*\*\*\*\*\*\*  Database: jiny  …….  …….  C:\Bitnami\wampstack-5.6.30-0\mysql\bin> |

쉘 명령을 이용하면, 직접 콘솔로 접속하여 SQL 쿼리 명령을 실행하지 않아도 되는 편리함이 있습니다.

**05.4 테이블중복**

====================

한 개의 동일한 데이터베이스 이름 그룹 안에서 테이블 명은 **고유한 이름**을 가지고 있습니다. 테이블의 이름은 중복하여 사용할 수 없습니다. 중복한다는 것은 중복된 테이블 파일을 생성을 하는 것과 같습니다. 만일 중복된 이름의 테이블을 생성할 경우에는 오류 메시지를 출력합니다.

데이터베이스 이름 그룹이 다르다면 동일한 이름의 테이블 명을 가질 수 있습니다. 즉, 테이블 이름의 중복 허용 원칙은 같은 데이터베이스 이름안에서만 제한됩니다.

테이블의 이름은 설치된 운영체제의 환경에 따라 **대소문자를 구분**하는 경우도 있습니다. 운영체제별로 틀린 것은 테이블명이 데이터베이스 디렉토리에 파일형태로 맵핑 되어 저장이 되기 때문입니다. 대소문자 구별은 시스템 환경설정을 통하여 제한할 수도 있습니다.

**05.4.1 중복 실험**

====================

그럼 일부러 기존에 존재하는 동일한 이름의 테이블 명을 실행해 보도록 해봅니다.

|  |
| --- |
| CREATE TABLE `members` (  `Id` int(11) NOT NULL AUTO\_INCREMENT,  PRIMARY KEY (`Id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8; |

콘솔 출력화면)

|  |
| --- |
| mysql> CREATE TABLE `members` (  -> `Id` int(11) NOT NULL AUTO\_INCREMENT,  -> PRIMARY KEY (`Id`)  -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  ERROR 1050 (42S01): Table 'members' already exists |

테이블의 이름이 같으면 오류를 발생합니다. 테이블의 컬럼 정보가 다르다고 해서 별도의 테이블로 인지하지는 않습니다. 테이블 명이 같으면 같은 테이블 중복으로 처리 됩니다.

**05.4.2 중복 확인**

====================

테이블을 생성할 때 테이블명의 중복은 오류로 처리됩니다. PHP를 통하여 테이블의 이름이 중복되어 있는지 확인하는 코드를 생성해 봅니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function isTables($tbname)  {  if ($tbname){  $dbname = $this->dbname;  $queryString = "show tables from $dbname like '$tbname'";  $this->msgEcho($queryString);  // 쿼리를 전송합니다.  if ($result = mysqli\_query($this->dbcon, $queryString)) {  if (mysqli\_num\_rows($result)) {  $row = mysqli\_fetch\_object($result);  $result->free();  $this->msgEcho($tbname."은 존재합니다.");  return true;  } else {  $this->msgEcho($tbname."이 없습니다. 생성이 가능합니다.");  return false;  }  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] 테이블이 존재합니다.");  }  } |

**예제파일) sql-13.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  $db->dbname = "jiny";  $tbname = "members123";  if ($db->isTables($tbname)){  echo $tbname." 존재하는 테이블 이름입니다.";  } else {    }  ?> |

출력화면)

mysql connected!

show tables from jiny like 'members123'

members123은 존재합니다.

members123 존재하는 데이터베이스 이름입니다.

**05.5 테이블 삭제**

====================

생성한 테이블을 삭제할 수 있습니다. DROP TABLE 쿼리명령을 통하여 기존에 존재하는 테이블을 삭제합니다.

**쿼리문법)**

|  |
| --- |
| DROP TABLES 테이블명; |

여러 개의 테이블을 삭제할 수도 있습니다.

**쿼리문법)**

|  |
| --- |
| DROP TABLE 테이블1, 테이블2; |

테이블을 삭제할 때는 매우 주의해야 합니다. 테이블을 삭제를 하게 되면 안에 들어있는 모든 데이터가 같이 삭제가 됩니다. 또한 삭제된 데이터들을 복구가 불가능하기 때문에 매우 신중하게 사용을 해야 합니다.

테이블 삭제 또한 데이터베이스 삭제에서 사용했던 IF EXISTS 를 같이 사용할 수 있습니다. 테이블이 존재하는 경우에만 삭제가 됩니다.

**쿼리문법)**

|  |
| --- |
| DROP TABLE IF EXISTS 테이블명; |

또는 여러 개의 테이블을 한번에 삭제할 수도 있습니다. 테이블 명을 콤마(,)로 구분하여 입력하면 됩니다.

**쿼리문법)**

|  |
| --- |
| DROP TABLE IF EXISTS 테이블1, 테이블2; |

**05.5.1 삭제 실습**

====================

쿼리 명령을 전송하여 테이블을 삭제해 봅니다.

|  |
| --- |
| DROP TABLES members; |

콘솔 출력화면)

|  |
| --- |
| mysql> DROP TABLES members;  Query OK, 0 rows affected (0.01 sec)  mysql> show tables;  Empty set (0.00 sec) |

기존에 생성되어 있는 "members" 테이블을 drop 명령어를 통하여 삭제가 성공되었다는 메시지를 확인할 수 있습니다.

**05.5.2 PHP 코드**

====================

PHP를 통하여 테이블을 삭제하는 메서드를 만들어 봅니다. 테이블을 삭제할 경우 데이터도 같이 삭제가 되는 위험이 있습니다. 만일, 안전한 삭제코드를 만든다고 한다면, 삭제하기 전에 데이터가 있지는 확인하는 루틴을 하나 더 삽입을 해도 좋을 듯 합니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function dropTable($tbname)  {  if ($tbname) {  $queryString = "DROP TABLES $tbname;";  $this->msgEcho($queryString);  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 테이블 삭제!");  // 객체 반환, 매서드체인  return $this;  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  }  } |

**예제파일) sql-14.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 데이터베이스 삭제합니다.  $tbname = "members123";  $db->dropTable($tbname);  ?> |

출력화면)

mysql connected!

DROP TABLES members123;

쿼리성공] DROP TABLES members123;

테이블 삭제!

**05.6 TRUNCATE**

====================

기존 테이블 삭제명령의 경우 단점은 테이블과 데이터를 모두 삭제 한다는 것 입니다. 만일 테이블은 남겨두고 안에 들어 있는 데이터만 모두 삭제 하고 싶을 때는 어떻게 해야 될까요? 테이블을 삭제하고 다시 신규 테이블 생성 명령을 실행할 수도 있습니다. 하지만 이것은 DB 시스템에 두번의 쿼리명령으로 불필요한 작업을 수행 시킬 수 있습니다.

이런 처리 요청을 위해 새로운 쿼리명령 TRUNCATE TABLE 을 사용할 수 있습니다.

**쿼리문법)**

|  |
| --- |
| TRUNCATE TABLE테이블명; |

TRUNCATE TABLE 명령은 기존의 테이블의 정보는 유지한 채로 안에 있는 데이터만 초기화 합니다.

**05.6.1 콘솔실습**

====================

기존 members 테이블에는 데이터가 존재합니다.

|  |
| --- |
| TRUNCATE TABLE members; |

콘솔 출력화면)

|  |
| --- |
| mysql> select \* from members;  +----+----------+-----------+----------+-------+---------+  | Id | LastName | FirstName | Address | City | Country |  +----+----------+-----------+----------+-------+---------+  | 1 | hojin | lee | shinchon | seoul | Korea |  +----+----------+-----------+----------+-------+---------+  1 row in set (0.00 sec)  mysql> **truncate table members;**  Query OK, 0 rows affected (0.02 sec)  mysql> select \* from members;  Empty set (0.00 sec) |

TRUNCATE TABLE members; 는 members 테이블 안에 있는 데이터만 삭제하고 테이블 자체는 남겨두고 있습니다. 다시 데이터를 조회해 보면 은 비어 있는 것을 확인 할 수 있습니다.

**05.6.2 PHP 코드**

====================

TRUNCATE TABLE 명령을 PHP 메서드로 추가하여 실습을 해보도록 합시다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function clearTable($tbname)  {  if ($tbname) {  $queryString = "TRUNCATE TABLE $tbname;";  $this->msgEcho($queryString);  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 테이블 초기화!");  // 객체 반환, 매서드체인  return $this;  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  }  } |

**예제파일) sql-15.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 데이터베이스 삭제합니다.  $tbname = "members1";  $db->clearTable($tbname);  ?> |

출력화면)

mysql connected!

TRUNCATE TABLE members1;

쿼리성공] TRUNCATE TABLE members1;

테이블 초기화!

**05.7 테이블 구조확인**

====================

우리는 앞에서 테이블의 목록과 테이블의 상세목록을 출력하는 학습과 실험을 해보았습니다. 하지만 테이블의 각각의 컬럼 정보와 속성까지 알기에는 부족합니다.

새로운 쿼리 명령어 DESC는 테이블의 상세 컬럼 정보까지 확인을 해 볼 수 있습니다.

**쿼리문법)**

|  |
| --- |
| DESC 테이블명; |

**05.7.1 콘솔 실습**

====================

DESC 쿼리명령은 지정한 테이블의 상세한 구조를 확인할 수 있습니다.

|  |
| --- |
| DESC members; |

콘솔 출력화면)

|  |
| --- |
| mysql> **desc members;**  +-----------+--------------+------+-----+---------+----------------+  | Field | Type | Null | Key | Default | Extra |  +-----------+--------------+------+-----+---------+----------------+  | Id | int(11) | NO | PRI | NULL | auto\_increment |  | LastName | varchar(255) | YES | | NULL | |  | FirstName | varchar(255) | YES | | NULL | |  | Address | varchar(255) | YES | | NULL | |  | City | varchar(255) | YES | | NULL | |  | Country | varchar(255) | YES | | NULL | |  +-----------+--------------+------+-----+---------+----------------+  6 rows in set (0.02 sec) |

DESC 쿼리명령은 테이블의 컬럼명, 데이터타입 및 속성까지 목록으로 출력합니다.

**05.7.2 PHP 코드**

====================

DESC 쿼리를 통하여 테이블의 컬럼목록을 출력하는 PHP 코드를 생성해 보도록 하겠습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| // 테이블 정보를 읽어 옵니다  public function descTable($tbname)  {  $queryString = "desc $tbname";  $this->msgEcho($queryString);  if ($result = mysqli\_query($this->dbcon, $queryString)) {  $rowss = "";  $row\_cnt = mysqli\_num\_rows($result);  for ($i=0;$i<$row\_cnt;$i++) {  $rowss[$i] = mysqli\_fetch\_object($result);  }  $result->free();  return $rowss;  }  } |

**예제파일) sql-16.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 테이블 구조를 확인합니다.  $tbname = "members";  if ($rowss = $db->descTable($tbname)) {  echo "tables fields = ". count($rowss) . "<br>";  for ($i=0;$i<count($rowss);$i++) {  echo $i."=";  print\_r($rowss[$i]);  echo "<br>";  }  }  ?> |

출력화면)

mysql connected!

desc members

tables fields = 8

0= stdClass Object ( [Field] => Id [Type] => int(11) [Null] => NO [Key] => PRI [Default] => [Extra] => auto\_increment )

1= stdClass Object ( [Field] => LastName [Type] => varchar(255) [Null] => YES [Key] => [Default] => [Extra] => )

2= stdClass Object ( [Field] => FirstName [Type] => varchar(255) [Null] => YES [Key] => [Default] => [Extra] => )

3= stdClass Object ( [Field] => Address [Type] => varchar(255) [Null] => YES [Key] => [Default] => [Extra] => )

4= stdClass Object ( [Field] => City [Type] => varchar(255) [Null] => YES [Key] => [Default] => [Extra] => )

5= stdClass Object ( [Field] => Country [Type] => varchar(255) [Null] => YES [Key] => [Default] => [Extra] => )

6= stdClass Object ( [Field] => manager [Type] => varchar(100) [Null] => YES [Key] => [Default] => [Extra] => )

7= stdClass Object ( [Field] => email [Type] => varchar(255) [Null] => YES [Key] => [Default] => [Extra] => )

**05.7.3 SHOW COLUMNS**

====================

DESC 명령어 이외에 SHOW COLUMNS 명령을 통하여 테이블의 구조를 확인 할 수도 있습니다.

**쿼리문법)**

|  |
| --- |
| SHOW COLUMNS FROM 테이블명; |

콘솔 출력화면)

|  |
| --- |
| mysql> **show columns from members;**  +-----------+--------------+------+-----+---------+----------------+  | Field | Type | Null | Key | Default | Extra |  +-----------+--------------+------+-----+---------+----------------+  | Id | int(11) | NO | PRI | NULL | auto\_increment |  | LastName | varchar(255) | YES | | | |  | FirstName | varchar(20) | YES | | NULL | |  | Address | varchar(255) | YES | | NULL | |  | address2 | varchar(255) | YES | | NULL | |  | City | varchar(30) | NO | | NULL | |  | Country | varchar(255) | YES | | NULL | |  | manager | varchar(100) | YES | | NULL | |  | email | varchar(255) | YES | | NULL | |  +-----------+--------------+------+-----+---------+----------------+  9 rows in set (0.09 sec) |

**05.8 테이블 데이터**

====================

데이터베이스의 테이블은 실제적인 데이터가 저장되어 있는 물리적인 파일입니다. 하지만 실제적인 테이블 파일을 직접 전근하여 조작을 할 수는 없습니다.

테이블의 데이터를 조작하기 위해서는 데이터를 처리할 수 있는 관련 SQL 명령 쿼리를 통하여 작업을 해야만 합니다.

* SELECT - 데이터를 읽고, 추출합니다.
* UPDATE - 데이터를 업데이트합니다.
* DELETE - 데이터를 삭제합니다.
* INSERT INTO - 새로운 데이터를 삽입합니다.
* CREATE DATABASE - 데이터베이스를 생성합니다.
* ALTER DATABASE - 데이터베이스를 수정합니다.
* CREATE TABLE - 새로운 테이블을 생성합니다.
* ALTER TABLE - 테이블을 수정합니다.
* DROP TABLE - 테이블을 삭제합니다.
* CREATE INDEX - index를 생성합니다. (search key)
* DROP INDEX - index를 삭제합니다.

앞으로 위와 관련된 SQL 문법들을 하나씩 배워 볼 것입니다.

**05.9 엔진설정**

====================

MYSQL은 데이터를 처리할 수 있는 다수의 데이터베이스 엔진이 존재합니다. 여러 개의 데이터 베이스 엔진은 MYSQL을 보다 유연하고 확장된 형태로 시스템을 운영할 수 있도록 환경을 제공합니다.

데이터베이스 엔진의 특성별로 장단점이 있습니다. 자신의 데이터의 성격에 맞는 엔진을 설정하여 사용을 할 수 있습니다.

MYSQL은 다음과 같은 엔진을 지원합니다:

* MyISAM
* InnoDB
* ISAM

**05.9.1 엔진확인**

====================

현재 테이블에 적용되어 있는 엔진을 확인할 수 있습니다.

**쿼리명령)**

|  |
| --- |
| show create table 테이블명; |

콘솔 출력화면)

|  |
| --- |
| mysql> show create table members;  +---------+------------------------------------------------------------------------------------------+  | Table | Create Table |  +---------+-----------------------------------------------------------------------------------------+  | members | CREATE TABLE `members` (  `Id` int(11) NOT NULL AUTO\_INCREMENT,  `LastName` varchar(255) DEFAULT NULL,  `FirstName` varchar(255) DEFAULT NULL,  `Address` varchar(255) DEFAULT NULL,  `City` varchar(255) DEFAULT NULL,  `Country` varchar(255) DEFAULT NULL,  `manager` varchar(100) DEFAULT NULL,  `email` varchar(255) DEFAULT NULL,  PRIMARY KEY (`Id`)  ) **ENGINE=InnoDB** AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8 |  +---------+------------------------------------------------------------------------------------------+  1 row in set (0.01 sec) |

현재 테이블이 InnoDB로 설정이 되어 있다는 것을 표시합니다.

**05.9.2 엔진변경**

====================

테이블의 엔진설정은 처음에 테이블을 생성을 할 때 같이 지정을 합니다. 하지만, 이렇게 설정한 테이블은 나중에 다른 엔진으로 변경을 할 수 있습니다.

**쿼리명령)**

|  |
| --- |
| ALTER TABLE 데이블명 ENGINE=엔진명 |

콘솔 출력화면)

|  |
| --- |
| mysql> alter table members ENGINE=MyISAM;  Query OK, 3 rows affected (0.04 sec)  Records: 3 Duplicates: 0 Warnings: 0 |

기존에 InnoDB로 설정이 되어 있던 엔진을 MyISAM으로 변경하였습니다.

**05.9.3 PHP 코드**

====================

PHP를 통하여 엔진 변경 메소드를 생성해 보도록 하겠습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| // 테이블 엔진을 설정합니다.  public function **setEngine**($tbname,$engine="InnoDB")  {  if ($tbname) {  $queryString = "ALTER TABLE $tbname ENGINE=$engine";    // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 엔진변경($engine)!");  // 객체 반환, 매서드체인  return $this;  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  }  } |

**예제파일) sql-17.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 테이블 엔진을 변경합니다.  $tbname = "members";  $db->setEngine($tbname,"MyISAM");  ?> |

출력화면)

mysql connected!

쿼리성공] ALTER TABLE members ENGINE=MyISAM

엔진변경(MyISAM)!

**05.10 테이블 최적화**

====================

테이블은 자료의 삽입과 수정, 삭제 작업이 자주 발생합니다. 이러한 잦은 데이터의 변경요구는 테이블의 실제 데이터의 구조를 엉키게 만들어 처리 속도를 저하합니다.

예전, 우리가 컴퓨터의 디스크 조각모음 기능을 실행했던 것처럼, 파일의 데이터 링크구조가 엉키게 됩니다. 만일 테이블에 가변폭 varchar 형태의 컬럼 데이터타입을 많이 사용을 했다고 한다며, 테이블은 불연속적인 공간은 더욱 늘어날 것입니다. 불연적인 공간이 증가할수록 테이블 데이터 구조는 낭비되고 속도가 저하 됩니다.

이런 경우 중간 중간 테이블 최적화 기능을 통하여 테이블의 불연속적인 공간을 정리해 주는 것이 좋습니다. 불연속적인 공간이 줄어들면 테이블의 파일 크기도 작아지고, 더 빠른 동작 처리가 가능해 집니다.

**쿼리문법)**

|  |
| --- |
| OPTIMIZE TABLE 데이블명 |

콘솔 실습)

|  |
| --- |
| mysql> optimize table members;  +--------------+----------+----------+----------+  | Table | Op | Msg\_type | Msg\_text |  +--------------+----------+----------+----------+  | jiny.members | optimize | status | OK |  +--------------+----------+----------+----------+  1 row in set (0.03 sec) |

OPTIMIZE TABLE은 작업 원리는 원본 데이터를 복사한 후에 데이터 정리작업을 진행합니다. 따라서 최적화 작업중에 검색과 같은 데이터 처리 작업은 유지할 수 있습니다. 정리 작업이 완료가 되면, 이전의 테이블은 삭제되고 정리된 새로운 테이블을 유지하게 됩니다.

주의할 점은 최적화 작업중에는 읽기만 가능하며, 새로운 자료의 삽입이나 업데이트가 잠시 지연됩니다.