jinyPHP 2권 - 007. 컬럼속성

**07. 속성**

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

테이블은 컬럼 데이터형 타입과 별개로 제약사항 속성을 추가할 수 있습니다. 추가된 제약사항 속성들은 컬럼의 데이터를 처리 하는데 별도의 규칙이 적용됩니다.

컬럼의 제약사항 속성은 반드시 사용해야 하는 것은 아닙니다. 옵션과 같은 성격을 뜁니다. 속성을 설정하게 되면 데이터의 무결성을 보완하고 테이블의 데이터 검색 속도 개선등의 이점들이 있습니다.

테이블을 생성을 할 때 기본적으로 많이 사용하는 몇가지의 속성들이 있습니다. Primary Key, Auto Increament 등이 있습니다. 제약사항 속성은 CREATE TABLE 명령을 통하여 테이블을 생성할 때 같이 설정을 합니다. 또는 ALTER TABLE 명령을 통하여 나중에 추가할 수 있습니다.

**쿼리문법)**

|  |
| --- |
| CREATE TABLE table\_name (  column1 datatype **constraint**,  column2 datatype **constraint**,  column3 datatype **constraint**,  ....  ); |

제약사항(constraint)는 컬럼의 데이터 타입설정 뒤에 위치합니다. MYSQL은 7가지의 제약사항 속성들을 지원합니다:

* NOT NULL - 컬럼 필드의 NULL 입력을 허용/방지 합니다.
* UNIQUE - 중복값 입력을 방지합니다.
* PRIMARY KEY - NOT NULL 와 UNIQUE 결합된 기본 데이터 키입니다.
* FOREIGN KEY - 다른 테이블의 값을 참조하여 연결합니다.
* CHECK - 데이터 입력 유효성을 설정합니다.
* DEFAULT - 컬럼 필드의 초기값을 설정합니다.
* INDEX - 빠른 접근을 위한 인덱스 입니다.

**07.1 제약사항 NOT NULL**

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

제약사항 “NOT NULL”은 컬럼의 NULL값에 대한 처리 여부를 지정하는 속성입니다. 보통 컬럼에 아무런 데이터가 입력이 되지 않으면 NULL 값으로 설정이 됩니다.

컬럼의 데이터에 반드시 들어가야 하는 필수 데이터값들은 “NOT NULL” 속성을 부여하는 것이 좋습니다. NOT NULL 제약사항이 설정된 컬럼은 데이터를 필수적으로 입력을 해야 되기 때문에 데이터값이 없는 경우 SQL 쿼리구문은 오류로 실행되지 않습니다.

NULL로 비어 있는 컬럼 값으로 인하여 발생할 수 있는 오류를 미리 사전에 방지할 수 있는 유용한 제약사항입니다.

**07.1.1 쿼리실습**

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

다음 예제는 “Customers” 테이블에서 “ID”, "LastName", "FirstName" 컬럼필드값의 NULL을 허용하지 않는 형태로 테이블을 생성을 합니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL,  LastName varchar(255) **NOT NULL**,  FirstName varchar(255) **NOT NULL**,  ); |

컬럼의 NULL 허용 및 방지는 테이블 생성후에도 ALTER TABLE 명령을 사용하여 수정을 할 수 있습니다.

|  |
| --- |
| ALTER TABLE 테이블명 CHANGE COLUMN `컬럼명` `컬럼명` 데이터 타입 NOT NULL |

또는

|  |
| --- |
| ALTER TABLE 테이블명 CHANGE COLUMN `컬럼명` `컬럼명` 데이터 타입 NULL |

**07.1.2 PHP 실습**

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

PHP 코드를 통하여 NOT NULL 제약사항 속성을 실습해 봅니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function constraintNOTNULL($tbname, $field, $datatype)  {  if ($tbname) {  $queryString = "ALTER TABLE $tbname CHANGE COLUMN `$field` `$field` $datatype NOT NULL";  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" NOT NULL 설정!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }    } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  return false;  }  }  public function constraintNULL($tbname, $field, $datatype)  {  if ($tbname) {  $queryString = "ALTER TABLE $tbname CHANGE COLUMN `$field` `$field` $datatype NULL";  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" NULL 설정!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }    } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  return false;  }  } |

**예제파일) attr-02.php**

|  |
| --- |
| <?php    include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();    $tbname = "members";    // 컬럼필드 NULL 허용하지 않습니다.  $db->constraintNOTNULL($tbname, "City", "varchar(30)");  // 컬럼필드 NULL 허용.  $db->constraintNULL($tbname, "FirstName", "varchar(20)");    ?> |

출력화면)

mysql connected!

쿼리성공] ALTER TABLE members CHANGE COLUMN `City` `City` varchar(30) NOT NULL

NOT NULL 설정!

쿼리성공] ALTER TABLE members CHANGE COLUMN `FirstName` `FirstName` varchar(20) NULL

NULL 설정!

NOT NULL 설정은 MYSQL 검색 및 처리 성능향상을 가지고 올 수 있습니다. 만일 모든 컬럼을 NOT NULL 로 설정하면 각각의 컬럼마다 1비트의 공간을 절약할 수 있습니다.

**07.2 기본값 속성(DEFAULT)**

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

제약사항 DEFAULT 속성은 컬럼에 입력되는 데이터의 값이 없을 경우 기본적으로 입력되는 초기값을 의미합니다. 특정한 컬럼에 데이터를 지정하지 않을 때 NULL로 설정이 되지 않고, 지정한 초기값으로 설정이 됩니다.

**07.2.1 쿼리실습**

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

테이블에서 City 속성은 기본값이 “Seoul”로 되어 있습니다. 만일 City 컬럼의 값이 없는 경우는 기본값 “Seoul”을 입력하라는 의미입니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  City varchar(255) **DEFAULT 'Seoul'**  ); |

기본값 속성은 다음과 같이 함수를 이용하여 설정을 할 수도 있습니다.

|  |
| --- |
| CREATE TABLE Orders (  Id int NOT NULL,  OrderNumber int NOT NULL,  RegOrder date DEFAULT **GETDATE()**  ); |

ALTER TABLE 명령을 통하여 기본값을 추가로 설정을 변경할 수 있습니다.

|  |
| --- |
| ALTER TABLE members ALTER City SET DEFAULT 'Seoul'; |

기본값을 삭제할 수도 있습니다.

|  |
| --- |
| ALTER TABLE members ALTER City DROP DEFAULT; |

**07.2.2 PHP 실습**

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

PHP 코드를 통하여 컬럼의 기본값 속성을 정의하는 실습합니다. 관련 메서드를 생성하여 소스에 적용합니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function constraintDEFAULT($tbname, $field, $default)  {  if ($tbname){  if($default){  $queryString = "ALTER TABLE $tbname ALTER $field SET DEFAULT '$default'";  } else {  $queryString = "ALTER TABLE $tbname ALTER $field DROP DEFAULT";  }  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE){  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 기본값 설정!");  // 객체 반환, 매서드체인  return $this;  } else {  $this->msgEcho("Error] ".$queryString);  }      } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  }  } |

**예제파일) attri-01.php**

|  |
| --- |
| <?php    include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();    $tbname = "members";  $field = "City";  $default = "Seoul";    // 속성 기본값을 설정합니다.  $db->constraintDEFAULT($tbname, $field, $default);    ?> |

출력화면)

mysql connected!

쿼리성공] ALTER TABLE members ALTER City SET DEFAULT 'Seoul'

기본값 설정!

**07.3 유일값(UNIQUE)**

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

제약사항 “UNIQUE”는 컬럼에 데이터의 중복여부를 처리하는 속성입니다. 컬럼에는 다양한 값들이 입력이 될 수 있는데, 특정한 컬럼의 경우 데이터를 구분하여 위하여 중복되지 않은 고유의 값만 처리를 해야 되는 경우도 많이 발생을 합니다.

컬럼의 UNIQUE 속성을 지정하게 되면 컬럼의 데이터 값은 중복을 허용하지 안습니다. 입력된 모든 데이터는 서로 다른 값을 가지고 있습니다. 만일 중복된 컬럼 데이터가 입력이 될 때, SQL 쿼리 명령은 오류를 발생합니다. 간단한 설정 값으로 중복으로 인하여 발생될 수 있는 오류를 사전에 방지할 수 있습니다.

프라이머리키(PRIMARY KEY) 제약사항 또한 자동적으로 유일값(UNIQUE) 속성과 같은 유일성이 보장되는 값들을 가집니다.

**07.3.1 쿼리실습**

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

UNIQUE 설정은 컬럼들을 선언한 뒤에 마지막에 별도의 추가 선언을 합니다. UNIQUE(컬럼명) 형태로 더 추가하면 됩니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  **UNIQUE (Id)**  ); |

또는 CONSTRAINT 키워드를 통하여 여러 컬럼을 UNIQUE 로 정의할 수 있습니다.

|  |
| --- |
| CREATE TABLE Persons (  ID int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  **CONSTRAINT UC\_members UNIQUE (Id,LastName)**  ); |

ALTER TABLE 명령을 통하여 기본값을 추가로 설정을 변경할 수 있습니다.

|  |
| --- |
| ALTER TABLE members ADD UNIQUE (Id); |

CONSTRAINT 키워드를 통하여 다수의 컬럼필드를 유일값으로 추할 수 있습니다.

|  |
| --- |
| ALTER TABLE members ADD CONSTRAINT UC\_members UNIQUE (Id,LastName); |

유일 값을 삭제할 수도 있습니다.

|  |
| --- |
| ALTER TABLE members DROP INDEX UC\_members; |

**07.3.2 PHP 실습**

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

PHP 코드를 통하여 UNIQUE 속성 처리에 대한 실습을 합니다. 관련 메서드를 추가하여 소스에 적용합니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function constraintUNIQUE($tbname, $field)  {  if ($tbname) {  $queryString = "ALTER TABLE $tbname ADD UNIQUE ($field)";  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 기본값 설정!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }    } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  return false;  }  }  public function constraintUNIQUE\_Drop($tbname, $field)  {  if ($tbname){  $queryString = "ALTER TABLE $tbname DROP INDEX `$field`";  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE){  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 기본값 설정!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }    } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  return false;  }  } |

**예제파일) attr-03.php**

|  |
| --- |
| <?php    include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();    $tbname = "members";  $field = "email";    // 컬럼필드 유일필드 설정  $db->constraintUNIQUE($tbname, $field);    ?> |

출력화면)

mysql connected!

쿼리성공] ALTER TABLE members ADD UNIQUE (email)

기본값 설정!

**07.4 프라이머리키(PRIMARY KEY)**

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

엑셀파일에서 좌측에 1,2,3,4,5 처럼 표시가 되는 숫자들을 보신적이 있을 것입니다. 테이블은 각각의 컬럼을 기준으로 다수의 데이터 열(Rows)을 가지고 있습니다.

테이블에 데이터를 저장할 때 각각의 데이터를 지정하기 위한 고유의 아이디 값이 필요로 합니다. 이러한 고유의 식별 값을 Primary Key로 지정을 합니다. 프라이머리키의 값은 앞에서 학습한 유일값(UNIQUE) 속성의 특징도 가지고 있습니다. 따라서 프라이머리 키의 값은 중복되지 않습니다. 또한 NOT NULL 속성도 가지고 있기 때문에 NULL을 허용하지 않고 반드시 값을 가지고 있습니다.

프라이머리 키는 테이블에 기본 키가 하나만 있을 수 있습니다. 기본 키는 하나 또는 여러 개의 필드로 구성 될 수 있습니다.

**07.4.1 쿼리 실습**

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

아래 예제는 정수형 Id 값을 프라이머리 키로 설정합니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  **PRIMARY KEY (Id)**  ); |

또는 다음처럼 프라이머리 키값을 여러개의 필드로 구성할 수도 있습니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  CONSTRAINT PK\_members PRIMARY KEY (**Id,LastName**)  ); |

위의 예제는 2개의 필드를 하나의 프라이머리 키로 설정을 합니다. 하지만, 프라이머리 키는 1개만 설정을 할 수 있기 때문에 CONSTRAINT 키워드를 통하여 PK\_members 키이름을 생성합니다.

프라이머리키는 ALTER 명령을 통하야 추가하거나 삭제할 수 있습니다.

|  |
| --- |
| ALTER TABLE members ADD PRIMARY KEY (Id); |

CONSTRAINT 키워드를 통하여 2개의 컬럼필드를 가지는 프라이머리 키를 추할 수 있습니다.

|  |
| --- |
| ALTER TABLE members  ADD CONSTRAINT PK\_members PRIMARY KEY (Id,LastName); |

2개의 필드를 가지는 프라이머리키 생성시 각각의 컬럼은 NULL값이 허용되지 않은 컬럼이어야 합니다.

**프라이머리키 삭제**

|  |
| --- |
| ALTER TABLE members DROP PRIMARY KEY; |

**07.5 CHECK**

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

앞에서 우리는 컬럼에 데이터를 삽입할 때 중복처리 및 필수, 초기값 등의 제약사항에 대해서 학습을 하였습니다. 제약사항 “CHECK”는 입력되는 데이터의 특정 조건을 필터링 하여 데이터를 처리하는 것입니다.

특정 값의 필터링은 보통 프로그램 코드 작성시에 조건 값을 판별하여 처리를 합니다. 하지만 간단한 로직은 CHECK 속성을 부과하여 데이터의 처리를 제한 할 수 있습니다. CHECK 설정된 조건을 벗어나게 되면 SQL 쿼리는 오류를 발생하고 처리되지 않습니다.

CEHCK 속성을 통하여 프로그램 로직과 이중으로 처리하는 것은 데이터의 안전한 유효성을 유지하는데 유용합니다.

**07.5.1 쿼리 실습**

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

CHECK 속성은 컬럼목록을 선언후에 별도로 추가 설정을 합니다. CEHCK(조건) 형태로 추가하시면 됩니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Age int,  **CHECK (Age>=18)**  ); |

CHECK 속성을 이용하여 나이를 입력하는 컬럼필드의 값을 18이상으로만 입력을 하도록 입력 제한을 합니다.

칼럼 한개만 check설정을 했을때는, 해당 칼럼필드만 check 값이 적용됩니다. 아니면, CONSTRAINT 키워드를 통하여 여러 컬럼의 조건을 설정할 수도 있습니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Age int,  Country varchar(255),  **CONSTRAINT CHK\_members CHECK (Age>=18 AND Country='Korea')**  ); |

또는 논리키워드를 이용하여 여러 컬럼의 체크조건을 설정합니다.

ALTER 명령을 통하여 추가하거나 삭제할 수 있습니다.

|  |
| --- |
| ALTER TABLE members ADD CHECK (Age>=18); |

CONSTRAINT 키워드를 통하여 다수의 컬럼필드를 변경할 수 있습니다.

|  |
| --- |
| ALTER TABLE members ADD CONSTRAINT CHK\_membersAge CHECK (Age>=18 AND Country='Korea'); |

삭제할 수도 있습니다.

|  |
| --- |
| ALTER TABLE members DROP CHECK CHK\_membersAge; |

**07.6 자동증가 (AUTO INCREMENT)**

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

자동증가 컬럼은 테이블의 새로운 데이터가 추가될 때 마다 컬럼의 값이 자동으로 증가 시키는 속성입니다. 자동증가 속성은 프라이머리키 속성과 같이 사용을 자주합니다. 또한 자동증가 컬럼은 UNIQUE 데이터가 들어가는 속성을 가집니다.

**07.6.1 쿼리 실습**

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

자동증가 속성은 컬럼의 타입이 INT, TINYINT, SMALLINT 형의 정수형 데이터 타입이어야 합니다.

|  |
| --- |
| CREATE TABLE members (  Id int NOT NULL **AUTO\_INCREMENT**,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  PRIMARY KEY (Id)  ); |

**자동증가값 초기화**

자동으로 증가되는 AUTO\_INCREAMENT 속성은 새로운 데이터를 신규로 입력할때 +1씩 증가합니다. +1씩 증가되는 카운트 값은 별도로 유지가 됩니다. 이전에 입력된 데이터를 삭제하더라도 자동 증가되는 값에는 변화가 없습니다. 자동 증가 값은 INSERT등의 데이터를 추가할 때만 변동이 됩니다.

만일 현재 유지되고 있는 자동 증가값을 변경하고 싶을 경우에는, ALTER 키워드를 통하여 자동증가 값을 초기화 할 수 있습니다.

|  |
| --- |
| ALTER TABLE members AUTO\_INCREMENT=0; |

**07.6.2 PHP 실습**

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

PHP 코드를 통하여 자동증가 속성을 처리해 보도록 합니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function autoIncrementClear($tbname,$inc=0)  {  if ($tbname) {  $queryString = "ALTER TABLE $tbname AUTO\_INCREMENT=$inc;";  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 자동증가값 초기화!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }    } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  return false;  }  } |

**예제파일) attr-04.php**

|  |
| --- |
| <?php    include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();    $tbname = "members";    // 테이블 자동증가 값을 초기화 합니다.  $db->autoIncrementClear($tbname,"100");    ?> |

출력화면)

mysql connected!

쿼리성공] ALTER TABLE members AUTO\_INCREMENT=100;

자동증가값 초기화!

100으로 초기화 하면, 앞으로 입력되는 자동증가 값은 100부터 시작합니다.

**07.7 색인(INDEX)**

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

테이블의 모든 데이터 열(Rows)의 자료는 처음부터 순차적으로 기록되지 않습니다. 즉, 정렬되어 입력이 되지 않습니다. 따라서 테이블에서 특정한 데이터를 찾기 위해서는 처음부터 끝까지 모든 자료를 검색해야 합니다. 전체 데이터를 검색하는 것은 매우 비효율 적입니다.

색인(index) 속성은 테이블에서 데이터를 검색할 때 속도를 개선하기 위해서 설정하는 속성입니다. 컬럼에 인덱스 속성을 설정하게 되면 빠른 자료 검색을 위해서 백과사전에서 색인을 만드는 것과 비슷합니다.

**07.7.1 색인생성**

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

색인생성은 CREATE INDEX 키워드를 이용하여 테이블/컬럼의 색인을 설정할 수 있습니다.

|  |
| --- |
| CREATE INDEX색인명 ON 테이블명 (컬럼명1, 컬럼명2...); |

대량의 데이터에 대해서 색인을 설정해 놓으면 빠른 검색 속도로 결과를 얻을 수 있습니다. 하지만 중복값이 많은 경우에는 색인설정이 오히려 더 성능에 방해될 수도 있습니다.

**07.2.2 색인확인**

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

SHOW 명령을 통하여 테이블의 설정된 색인(index)를 확인할 수 있습니다.

|  |
| --- |
| SHOW INDEX FROM 테이블명; |

|  |
| --- |
| mysql> **show index from members;**  +---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+  | Table | Non\_unique | Key\_name | Seq\_in\_index | Column\_name | Collation | Cardinality | Sub\_part | Packed | Null | Index\_type | Comment | Index\_comment |  +---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+  | members | 0 | PRIMARY | 1 | Id | A | 7 | NULL | NULL | | BTREE | | |  +---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+  1 row in set (0.00 sec) |

**07.2.3 색인삭제**

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

설정한 색인을 삭제할 수도 있습니다.

|  |
| --- |
| ALTER TABLE table\_name DROP INDEX index\_name; |

**07.8 외래키 (FOREIGN KEY)**

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

관계형 데이터베이스의 특징은 데이터를 중복과 저장 공간의 효율성을 위해서 여러 개의 테이블에 분산하여 저장을 합니다. 또한 분산되어 있는 테이블의 데이터를 간에 관계를 지정하여 복합적인 처리를 할 수 있도록 합니다.

외래키는 테이블들 간의 관계를 지정하는 링크 연결고리와 같습니다. 외래키는 FOREIGN KEY라고 부릅니다. 외래키는 관계형 데이터베이스에서 설계하는 데이터 모델링과도 중요한 밀접관계를 가지고 있습니다.

**07.8.1 외래키 설정**

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

FOREIGN KEY는 링크 연결 고리로 다른 테이블의 PRIMARY KEY의 값을 가집니다

|  |
| --- |
| CREATE TABLE Orders (  OrderID int NOT NULL,  OrderNumber int NOT NULL,  PersonID int,  PRIMARY KEY (OrderID),  **FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)**  ); |

FOREIGN KEY 제약 조건의 이름 지정을 허용하고 여러 열에 FOREIGN KEY 제약 조건을 정의하려면 다음 SQL 구문을 사용합니다.

|  |
| --- |
| CREATE TABLE Orders (  OrderID int NOT NULL,  OrderNumber int NOT NULL,  PersonID int,  PRIMARY KEY (OrderID),  CONSTRAINT FK\_PersonOrder FOREIGN KEY (PersonID)  REFERENCES Persons(PersonID)  ); |

**07.8.2 외래키 수정**

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

외래키는 ALTER TABLE 명령을 통하여 추가하거나 삭제할 수 있습니다.

|  |
| --- |
| ALTER TABLE Orders ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID); |

또는 CONSTRAINT 키워드를 통하여 다수의 컬럼필드를 변경할 수 있습니다.

|  |
| --- |
| ALTER TABLE Orders ADD CONSTRAINT FK\_PersonOrder  FOREIGN KEY (PersonID) REFERENCES Persons(PersonID); |

**07.8.2 외래키 삭제**

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

설정한 외래키는 ALTER TABLE 명령을 통하여 삭제가 가능합니다.

|  |
| --- |
| ALTER TABLE Orders DROP FOREIGN KEY FK\_PersonOrder; |