jinyPHP 2권 - 024. 트랜잭션

**24. 트랜잭션**

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

트랜잭션을 이해를 하기 위해서는 먼저 내부적으로 MYSQL의 쿼리입력과 처리에 대한 추가적인 지식이 필요로 합니다. SQL 쿼리 명령을 입력하게 되면, 입력 처리와 동시에 쿼리의 명령은 실행이 됩니다. 모든 쿼리 명령들은 실행과 동시에 실제적인 데이터의 변경 작업을 하게 됩니다.

우리는 앞에서 프로시저에 대해서 간단하게 학습을 하였습니다. 프로시저는 복수의 쿼리를 실행 하는데 있어서 중간의 쿼리의 오류를 방지하기 위해서 일괄 처리 하는 명령입니다. 트랜잭션은 좀더 심화적으로 실행도중에 발생될 수 있는 쿼리를 임시로 실행하고 이를 실제적인 데이터베이스에 반영을 하는 것입니다.

트랜잭션에서는 커밋이라는 개념이 등장합니다. 모든 쿼리 작업등은 임시로 처리하고 커밋을 통하여 실제적으로 데이터 반영을 하게 됩니다. 트랜잭션은 SQL 명령실행시 오류 또는 시간이 많이 걸리는 작업이 사용을 하면 좋습니다.

트랜잭션 기능은 MYSQL 버전과 달리 엔진에 영향을 받습니다. 트랜잭션을 적용하기 위해서는 DB 엔진을 **innoDB** 엔진으로 설정합니다.

**24.1 커밋 모드**

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

트랜잭션에는 커밋이라는 명령을 같이 사용합니다. 커밋은 임시로 실행처리한 쿼리 명령을 실제적인 데이터베이스에 반영을 처리 하는 명령 입니다.

기본적으로 데이터베이스는 커밋 동작에 대하여 자동 처리 되어 있습니다. 자동처리한 현재와 같이 SQL 쿼리를 입력하고 실행하게 되면 바로 실시간으로 데이터를 반영하여 처리를 하는 것입니다.

트랜잭션으로 작업을 하기 위해서는 자동 처리되는 커밋을 수동으로 변경을 해야 합니다. 수도 커밋모드 상테에서 실제적인 데이터 쿼리 작업을 실행합니다. 쿼리의 실행이 정상적이라고 판단이 되어쓸 때 COMMIT 명령을 통하여 작업을 실제 데이터베이스에 반영을 합니다.

**쿼리명령)**

|  |
| --- |
| START TRANSACTION; |

START TRANSACTION; 은 현재 자동 커밋 모드상태에서 트랜잭션을 시작하겠다는 명령입니다. 트랜잭션 시작을 하게 되면 자동커밋 상태에서 수동커밋 상태로 전환이 됩니다.

**24.1.1 쿼리 실습**

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

트랜잭션 시작을 통하여 수동커밋 모드 상태로 전환을 합니다.

|  |
| --- |
| mysql> START TRANSACTION;  Query OK, 0 rows affected (0.00 sec) |

**24.1.2 PHP 실습**

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

PHP 코드를 통하여 트랜잭션 모든 전환 메서드를 생성해 봅니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function transaction()  {  $queryString = "START TRANSACTION";    // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE){  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 트렌젝션 시작!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }  } |

**24.2 커밋**

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

트랜잭션 시작으로 커밋 모드를 수동 상태로 전환을 합니다. 수동 모드에서도 모든 쿼리 명령들을 실행을 할 수 있습니다 하지만 임시로 처리가 되기 때문에 실제적인 DB 시스템에 반영은 되지 안습니다. 수동상태는 임시모드의 쿼리 작업들 입니다.

수동 커밋 모드에서 처리된 쿼리들은 COMMIT 명령을 입력하게 되면 실제적인 데이터 베이스 시스템에 반영처리를 하게 됩니다.

**쿼리명령)**

|  |
| --- |
| COMMIT; |

DB 시스템이 커밋 명령을 입력 받으면 트랜잭션 동작을 종료합니다. 그리고 수동커밋 모드에서 자동커밋 모드 상태로 돌아갑니다.

**24.2.1 쿼리 실습**

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

수동 커밋모드 상태에서 작업한 쿼리를 DB 시스템에 반영을 합니다.

|  |
| --- |
| mysql> COMMIT;  Query OK, 0 rows affected (0.00 sec) |

**24.2.2 PHP 실습**

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

PHP 코드를 통하여 커밋 명령을 처리하는 메소드를 생성해 봅니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function commit()  {  $queryString = "COMMIT";    // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 작업내용 커밋!");  } else {  $this->msgEcho("Error] ".$queryString);  }  } |

**예제파일) trans-01.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 트렌젝션시작  if ($db->transaction()){    // 데이터 전체를 삭제합니다.  $tbname = "members\_all";  $db->deleteAll($tbname);  // 작업내용 실제적용  if ($db->commit()){  echo "모든 데이터가 적용 커밋 하였습니다.<br>";  } else {  echo "커밋을 실패하였습니다.";  }  } else {  echo "트렌젝션 모드로 전환할 수 없습니다.";  }  ?> |

출력화면)

mysql connected!

쿼리성공] START TRANSACTION

트렌젝션 시작!

쿼리성공] DELETE FROM members\_all

데이터 전체 삭제!

쿼리성공] COMMIT

작업내용 커밋!

모든 데이터가 적용 커밋 하였습니다.

**24.3 롤백**

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

트랜잭션을 이용하는 목적은 쿼리의 작업을 수행하는 동안 발생된 쿼리작업들을 되돌릴 수 있기 때문입니다. 작성한 쿼리 명령들이 즉시 데이터베이스에 적용이 된다고 생각해 봅니다. 한 개의 쿼리 명령이라면 쿼리를 수정하여 처리를 다시 하면 됩니다. 하지만 연관된 여러 쿼리를 복잡하게 처리 해야 하는데 중간에 쿼리 작업 하나가 잘못 되었다고 생각해 봅니다. 이전의 작업들이 이미 적용이 되어서, 다시 되돌리기 힘들 것입니다.

트랜잭션의 수동커밋은 작업한 쿼리 명령들이 임시로 처리가 되기 때문에 트랜잭션을 처리하기 이전의 상태로 되돌릴 수 있습니다.

**쿼리문법)**

|  |
| --- |
| ROLLBACK; |

롤백 명령은 트랜잭션에서 처리한 작업을 되돌리기 합니다.

**24.3.1 쿼리 실습**

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

트랜잭션 모드로 변경후에 쿼리 작업을 실습해 봅니다. 먼저 트랜잭션 모드로 작업하기 전의 상태를 미리 확인을 합니다.

|  |
| --- |
| // 기존 테이블의 데이터를 확인합니다.  mysql> select \* from members;  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  | Id | LastName | FirstName | Address | City | Country | manager | email |  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  | 1 | hojin | lee | shinchon | seoul | korea | infohojin | hojin@jinyphp.com |  | 2 | jiny | NULL | NULL | NULL | korea | infohojin | jiny@jinyphp.com |  | 3 | james | kim | youngdun-po | seoul | korea | info | james@jinyphp.com |  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  3 rows in set (0.01 sec) |

트랜잭션 시작을 명령합니다. 트랜잭션 처리가 되면 수동 커밋 상태로 전환이 됩니다.

|  |
| --- |
| // 트랜잭션 모드를 시작합니다.  mysql> START TRANSACTION;  Query OK, 0 rows affected (0.00 sec) |

트랜잭션 상태에서 일부러 실수를 만들어 봅니다. 원래는 데이터 하나만 삭제를 해야 되는데, 조건을 설정하기 않아서 회원 테이블 전체의 데이터가 삭제가 되었습니다.

|  |
| --- |
| // 테이블의 데이터를 삭제합니다.  mysql> delete from members;  Query OK, 3 rows affected (0.00 sec)  // 삭제된 결과를 확인합니다.  mysql> select \* from members;  Empty set (0.00 sec) |

아직 COMMIT 명령을 실행하지 않았기 때문에 실제적인 데이터 적용은 되지 안았습니다. 다행으로 트랜잭션의 작업들은 되돌리기가 가능합니다.

|  |
| --- |
| // 롤백합니다.  mysql> ROLLBACK;  Query OK, 0 rows affected (0.01 sec) |

되돌리기 작업을 한 후에 다시한번 회원테이블을 확인해 봅니다.

|  |
| --- |
| // 삭제된 데이터가 롤백됩니다.  mysql> select \* from members;  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  | Id | LastName | FirstName | Address | City | Country | manager | email |  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  | 1 | hojin | lee | shinchon | seoul | korea | infohojin | hojin@jinyphp.com |  | 2 | jiny | NULL | NULL | NULL | korea | infohojin | jiny@jinyphp.com |  | 3 | james | kim | youngdun-po | seoul | korea | info | james@jinyphp.com |  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  3 rows in set (0.00 sec)  mysql> |

데이터가 정상적으로 돌아 왔습니다. 롤백기능은 SQL 작업들을 되돌릴 수 있지만, 일부 명령들은 적용이 되지 않습니다.

* DROP DATABASE
* DROP TABLE
* DROP 명령들
* ALTER TABLE

은 롤백기능이 지원하지 않습니다. 위의 명령들은 입력시 자동으로 커밋됩니다.

**24.3.2 PHP 실습**

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

PHP 코드를 통하여 롤백 메서드를 만들어 봅니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function rollback()  {  $queryString = "ROLLBACK";    // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 작업내용 켄슬(롤백)!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }  } |

**예제파일) trans-02.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 트렌젝션시작  if ($db->transaction()) {    // 데이터 전체를 삭제합니다.  $tbname = "members\_all";  $db->deleteAll($tbname);  // 작업내용 롤백  // 위의 모든 데이터 작업을 취소합니다.  $db->rollback();  } else {  echo "트렌젝션 모드로 전환할 수 없습니다.";  }  ?> |

출력화면)

mysql connected!

쿼리성공] START TRANSACTION

트렌젝션 시작!

쿼리성공] DELETE FROM members\_all

데이터 전체 삭제!

쿼리성공] ROLLBACK

작업내용 켄슬(롤백)!

**24.4 환경설정 변경**

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

트랜잭션의 명령은 DB시스템의 커밋상태를 임시로 변경을 하게 됩니다. 기본적으로 커밋상태는 자동모드 입니다.

트랜잭션 시작 명령 이외에도 기본적으로 동작하는 모드를 변경을 할 수도 있습니다.

**24.4.1 자동커밋**

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

설정명령을 통하여 자동커밋 상태를 활성화 합니다.

**자동커밋 OFF)**

|  |
| --- |
| SET AUTOCOMMIT=0; |

콘솔에서 명령을 입력화여 커밋 모드를 변경해 보도록 합니다. 0의 값은 자동 커밋 모드 상태를 의미합니다.

|  |
| --- |
| mysql> SET AUTOCOMMIT=0;  Query OK, 0 rows affected (0.01 sec) |

**24.4.2 수동커밋**

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

설정명령을 통하여 수동커밋 상태로 활성화 합니다.

**자동커밋 ON)**

|  |
| --- |
| SET AUTOCOMMIT=1; |

콘솔에서 명령을 입력화여 커밋 모드를 변경해 보도록 합니다. 1의 값은 수동 커밋 모드 상태를 의미합니다.

|  |
| --- |
| mysql> SET AUTOCOMMIT=1;  Query OK, 0 rows affected (0.00 sec) |

**24.4.3 PHP 실습**

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

PHP 코드를 통하여 커밋 모드 상태를 변경하는 메서드를 만들어 봅니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function autoCommit($mode)  {  if ($mode == true) {  $queryString = "SET AUTOCOMMIT=1;";  } else {  $queryString = "SET AUTOCOMMIT=0;";  }  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  if ($mode == true) {  $this->msgEcho(" 자동커밋 상태전환!");  } else {  $this->msgEcho(" 자동커밋 OFF!");  }    return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }  } |