정보학연구 제12권 제2호 2009년 6월(pp.49-55)
The Journal of Information Technology. Vol.12. No.2

단일 Database Field를 이용한 PHP 무한 File Upload 구현

황 기 환* · 변 기 환** · 이 지 영***

목 차

Ⅰ. 서 론

클라이언트에서 서버로 파일을 업로드한 후 업로드 된 파일을 게시판이나 Intranet 자료실에서 사용하기 위해서는, RFC 1867 표준문서에 맞는 HTML POST 폼을 사용해 웹브라우저를 통해 업로드 된 파일을 웹 프로그램(PHP, ASP, JSP, CGI)으로 파일을 특정 폴더에 저장하고, 해당 파일명은 DB에 저장한 후 파일이 저장된 경로와 DB에 저장된 파일명을 합하여 전체 파일 경로를 만든 후 링크를 설정한다[1]-[3].

DB 파일명을 저장하기 위해서는 파일명을 저장하기 위한 필드가 필요한데, 단일필드방식은 하나의 필드에 하나의 파일이름이 저장되므로, 업로드 할 파일 개수만큼 필드를 생성해야 한다. 따라서 기존 다중필드를 이용한 파일업로드 방식은 사용자가 업로드 하고자 하는 파일 개수만큼 업로드가 불가능 하며, 통상적으로 프로그램을 만든 프로그래머가 지정한 개수만큼

* 경북전문대학 컴퓨터정보과 조교수
** 경북전문대학 컴퓨터정보과 겸임교수
*** 세명대학교 컴퓨터학부 교수

만 업로드가 가능하다. 이는 필드의 개수를 고정으로 할당해야 하므로 파일이 없는 무효 필드에 대해서는 효율성에서 비합리적이다. 또한 고정된 필드 개수로 Upload 파일 개수가 한정되기 때문에 파일의 개수가 많은 경우 여러 번 Upload 처리를 실행해야하는 경우가 발생된다. 또한 데이터베이스 테이블에서 다수개의 필드를 사용하는 경우, 파일 명을 삽입(Insert)하거나 업 데이터(Update), 또는 삭제(Delete)할 때 파일 명의 각각의 필드(Field)를 모두 검색해야하는 제약을 갖는다.

본 연구에서 다중 업로드 방식은 데이터베이스(Database) 테이블(table)에 파일 이름을 할당할 필드(Field)를 여러 개의 필드를 사용하지 않고, 하나의 단일 필드에 구분자를 이용하여 파일 개수에 관계없이 유동적으로 파일이름을 저장하는 방법을 제안한다.

II. 본론

기존의 다중필드 다중업로드 방식은 [그림1]과 같이 업로드 하려는 파일 개수만큼 필드를 생성해야 하며, 한 번에 업로드 가능한 파일의 수는 해당 필드의 개수와 같다.

필드	종류	Collation	보기	Null	기본값	추가	실행
uid	int(11)			아니오		auto_increment	
file_0	varchar(50)	utf8_general_ci		아니오			
file_1	varchar(50)	utf8_general_ci		아니오			
file_2	varchar(50)	utf8_general_ci		아니오			
file_3	varchar(50)	utf8_general_ci		아니오			
file_4	varchar(50)	utf8_general_ci		아니오			
file_5	varchar(50)	utf8_general_ci		아니오			
file_6	varchar(50)	utf8_general_ci		아니오			
file_7	varchar(50)	utf8_general_ci		아니오			
file_8	varchar(50)	utf8_general_ci		아니오			

그림 1. 다중필드 다중업로드 방식의 DB schema

제안하는 새로운 방식의 DB schema는 [그림2]와 같이 파일명을 저장하는 필드가 단일필드로 구성된다.

필드	종류	Collation	보기	Null	기본값	추가	실행
uid	int(11)			아니오		auto_increment	
file	text	utf8_general_ci		아니오			

그림 2. 단일필드 다중업로드 방식의 DB schema

본 연구에서 제안한 단일 DB필드를 이용한 다중 파일 업로드 방식은 기존의 방식에 비해 다음과 같은 유효성을 갖는다.

① 게시판이나 사내의 Intranet 자료실 등의 데이터처럼 한 번에 업로드하려는 파일의 개수가 몇 개인지 예측할 수 없는 유동적인 경우, 단일필드에 단지 구분자로 파일 명을 구분하므로 파일 개수를 고려해 필드의 개수를 고정하는 기존의 방식과는 달리 파일 명이 없는 무효 필드가 발생되지 않는다. 또한 Upload할 파일 개수가 많은 경우에도 한 번에 파일을 업로드 할 수 있다.

② 제안된 단일 필드 방식은 파일 명을 삽입(Insert), 업 데이터(Update) 또는 삭제(Delete)할 때, 단일 필드를 이용함으로 기존의 다수의 필드를 모두 검색해야하는 방식과는 달리 파일 필드를 한번만 검색(Select) 하면 되므로 서버의 부하를 줄임은 물론 네트워크 전체의 트래픽을 향상시킬 수 있는 합리적인 방법이다.

2.1 업로드 파일선택

파일업로드는 RFC 1867 규칙에 따라 HTML 폼 태그인 content type이 "multipart/form-data" 이고 post method 일 때 input type method를 file로 지정하면 웹브라우저에 로컬파일을 선택할 수 있는 형태로 표현 된다.

```
<form name='form_1' method='post' action='save.php' enctype='multipart/form-data'
```

그림 3. 본 연구코딩에 사용된 form tag

본 연구 코딩에서는 그림4]와 같이 업로드 될 파일의 개수를 업로드를 하는 사용자가 선택할

수 있도록 하였다.

그림 4. 첨부가능 한 파일을 사용자가 선택하는 화면

사용자가 허용한 첨부가능 한 파일개수는 $board_code[pds_num] 변수에 저장되며, 그림6과 같이 $board_code[pds_num]의 값에 따라 반복하여 [그림5]와 같이 로컬 파일 불러오기 화면을 표현하게 된다.

```
for($i = 0; $i < $board_code[pds_num]; $i++)
{
    $up_file_form .= "<input type='file' name='upfile[]' style='width:100%'
}
```

그림 5. 로컬 파일선택 화면을 출력하기 위한 PHP 코딩

클라이언트는 사용자의 요구만큼 [그림6]과 같이 파일 불러오기 화면을 출력하고 파일과 관련된 모든 아이템은 “upfile[]'라는 변수에 저장되어 서버로 전송된다.

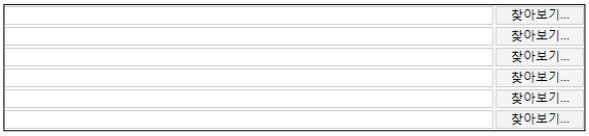


그림 6. [그림5]에 의해 표현된 로컬 파일선택화면

2.2 업로드처리

2.2.1 신규 등록 시

그림8과 같이 클라이언트의 웹브라우저에서 보내온 파일 변수 “upfile"를 "sizeof ($upfile)" 함수에 대입하여 업로드 된 파일의 개수를 반환 받아 개수만큼 루프를 돌면서 파일명에 공백을 원하는 기호로 치환하고 업로드가 불가능한 파일은 리턴을 시키고, 업로드가 될 폴더가 존재하지 않는다면 ” mkdir()' 함수를 사용하여 폴더를 생성한 다음 파일을 업로드 한다.

파일이 업로드 된 후 database의 file 필드에는 [그림8]과 같이 파일이름이 기호 “^” 로

```
for($ii = 0; $ii < sizeof($upfile); $ii++)
{
    if($upfile_name[$ii] != "")
    {
        // 파일 확장자
        $ext[$ii] = substr($upfile_name[$ii],strlen($upfile_name[$ii])-3);

        // 공백은 "_"로 변환
        $upfile_name[$ii] = str_replace(" ", "_", $upfile_name[$ii]);

        if(@eregi($ext[$ii], "HTM|HTML|PHP|PHP3|PHP4|CGI|INC|html|php3|php|inc|htm|cgi|asp"))
            // 업로드가 불가능한 파일처리

        if(@file_exists("./Db/$db/UPfile/$upfile_name[$ii]"))
            // 이미 파일이 해당 폴더에 존재하는 경우 처리

        if(!is_dir("./Db/$db/UPfile"))
            //업로드 디렉토리가 없으면 생성

        if(!move_uploaded_file($upfile[$ii], "./Db/$db/UPfile/$upfile_name[$ii]"))
            // 파일을 업로드 하지 못했을 경우 처리

    }
    $file_name .= $upfile_name[$ii] == "" ? "" : $upfile_name[$ii]."^";
}
```

그림 7. 단일필드 다중 업로드 방식의 파일업로드

구분되어 저장 된다.

			uid	file
☐	✎	✕	1	File_1_1.JPG^File_2_1.JPG^File_3_1.JPG^File_4_1.JP
☐	✎	✕	2	File_1_2.JPG^File_2_2.JPG^File_3_2.JPG^File_4_2.JP
☐	✎	✕	3	File_1_3.JPG^File_2_3.JPG^File_3_3.JPG^File_4_3.JP

그림 8. DB에 저장된 파일이름

2.2.2 수정

기존에 등록된 파일 이름을 $row->file 변수에 반환 받은 다음, split() 함수로 “^” 기호를 구분하여 파일이름을 "$file_array" 변수에 배열로 저장하고 삭제하려는 파일의 아이템 번호를 체크박스로 전달한다.

사용자의 요구로 인한 수정인 경우 세 가지 상황을 고려해야 한다. 첫 번째 파일의 변화 없이 수정하는 경우, 두 번째 기존에 등록된 파일을 다른 파일로 대체하는 경우, 세 번째 기존에 등록된 파일을 삭제하는 경우이다.

이 세 가지 상황을 고려하여 세 개의 변수를 서버에 전달되도록 한다. 첫 번째 상황은 파일 수정이 없으므로 DB를 수정하지 않는다. 두 번째 상황인 경우 새로운 파일을 등록할 수 있도록 upfile[] 변수를 사용하였으며, 기존 파일삭제를 위하여 vfile[] 변수에 value 값으로 원본 파일명을 같이 전송하여, 먼저 기존 파일을 삭제한 후 새로 등록 된 파일명으로 대체 되도록

하였다.

그림 9. 수정화면을 출력하기 위한 PHP 코딩

마지막으로 해당 파일을 삭제하는 경우에는 아이템 리스트와 원본 파일명을 전송하면 해당 파일명으로 폴더를 검사하고 파일이 존재하는 경우 해당 파일을 삭제하고 DB에 저장할 파일명은 없는 것으로 처리하였다.

그림 10 등록된 파일을 수정을 하기위한 PHP 코딩

III. 실험

본 연구에서 제안한 단일 DB 필드를 이용한 다중 파일 업로드 방식의 유효성을 검증하기 위해 다음과 같은 실험 조건으로 업로드 처리 속도와 검색 속도를 측정하여 기존의 다중 DB 필드를 이용한 다중 파일 업로드 방식과 비교 분석하였다.

3.1 업로드 속도 실험

APM(Apache,Php,MySQL)을 Local 컴퓨터에 설치하여 Local 컴퓨터로 업로드 하는 과정을 실험하였다. 실험을 위해 파일 하나가 1Mb인 이미지파일을 한꺼번에 10개씩 업로드 하는 실험 조건으로 업로드 속도를 측정하였다.

속도측정을 위하여 업로드가 시작될 때 $T1변수에 microtime() 함수를 사용하여 시작 시간을 체크하고, 업로드가 끝날 때 $T2 = microtime(true)를 실행하여 $T1에서 $T2값을 뺀 시간을 측정하였다. 단일필드 다중로드 속도측정 실험을 위한 PHP 코딩은 [그림11]과 같이 하였으며, 다중필드 다중업로드 속도측정 실험을 위한 PHP 코딩은 [그림12]와 같이 하였다.

```php
// 처음 시작시간
$T1 = microtime(true);

for($i=0; $i < count($upfile); $i++)
{
    if($upfile[$i] != "" || $upfile[$i] != "none")
    {
        $Filename = $upfile_name[$i];
        $dir  = "./test_2/".$Filename;

        move_uploaded_file($upfile[$i],$dir);

        $file_name .= $upfile_name[$i] == "" ? "" : $upfile_name[$i].
    }
}
$sql = "insert into test_2 (file) values ('$file_name')";
mysql_query($sql);

// 업로드가 종료된 시간
```

그림 11. 단일필드 다중업로드 업로드속도 측정을 위한 PHP 코딩

```php
// 처음 시작시간
$T1 = microtime(true);
for($i=0; $i < count($upfile); $i++)
{
    if($upfile[$i] != "" || $upfile[$i] != "none")
    {
        $Filename = $upfile_name[$i];
        $dir  = "./test_1/".$Filename;

        move_uploaded_file($upfile[$i],$dir);
        if($i == 0)
            $file_name .= "'".$Filename."'";
        else
            $file_name .= ",'".$Filename."'";
    }
}
$sql = "insert into test_1 (file_1,file_2,file_3,file_4,file_5,fil
                            file_7,file_8,file_9,file_10)
                    values
                            ($file_name)";
mysql_query($sql);
```

그림 12. 다중필드 다중업로드 실험을 위한 PHP 코딩

5회에 걸쳐 연속 실험한 업로드 속도측정 결과는 다음과 같다.

	단일 DB 필드(Sec)	다중 DB 필드(Sec)	개선률
1회	0.008457	0.036854	430%
2회	0.005353	0.013871	250%
3회	0.004866	0.012300	250%
4회	0.005182	0.017209	330%
5회	0.003093	0.010497	330%

표 1. 업로드 속도 측정

3.2 검색 속도 실험

본 실험조건으로 기존의 다중 DB 필드방식은 Index 1개와 파일 이름 필드 10개를 갖는 9999개의 레코더로 기록된 데이터로 구성하였으며, 제안한 단일 DB 필드방식은 Index 1개와 1개의 파일 이름 필드를 갖는 9999개의 레코더로 구성하여 사용하였다.

uid	file_1	file_2	file_3
9995	File_1_9991.JPG	File_2_9991.JPG	File_3_9991.JPG
9996	File_1_9992.JPG	File_2_9992.JPG	File_3_9992.JPG
9997	File_1_9993.JPG	File_2_9993.JPG	File_3_9993.JPG
9998	File_1_9994.JPG	File_2_9994.JPG	File_3_9994.JPG
9999	File_1_9995.JPG	File_2_9995.JPG	File_3_9995.JPG

그림 13. 기존 방식의 총 9999개 레코더를 구성

uid	file
9995	File_1_9995.JPG^File_2_9995.JPG^File_3_9995.JPG^Fi...
9996	File_1_9996.JPG^File_2_9996.JPG^File_3_9996.JPG^Fi...
9997	File_1_9997.JPG^File_2_9997.JPG^File_3_9997.JPG^Fi...
9998	File_1_9998.JPG^File_2_9998.JPG^File_3_9998.JPG^Fi...
9999	File_1_9999.JPG^File_2_9999.JPG^File_3_9999.JPG^Fi...

그림 14. 제안한 방식의 총 9999개 레코더를 구성

검색 처리속도 측정을 위해 임의로 총 9999개의 레코드를 등록하고 이중 지정된 범위의 레코드를 검색할 때 처리되는 시간을 phpMyAdmin을 사용하여 해당 범위를 선택하였을 때 출력되는 시간을 측정하였다.

검색 레코드	단일 DB 필드(Sec)	다중 DB 필드(Sec)
0~29	0.0006	0.0006
30~59	0.0006	0.0006
60~89	0.0006	0.0079
90~119	0.0077	0.0070
120~149	0.0065	0.0089

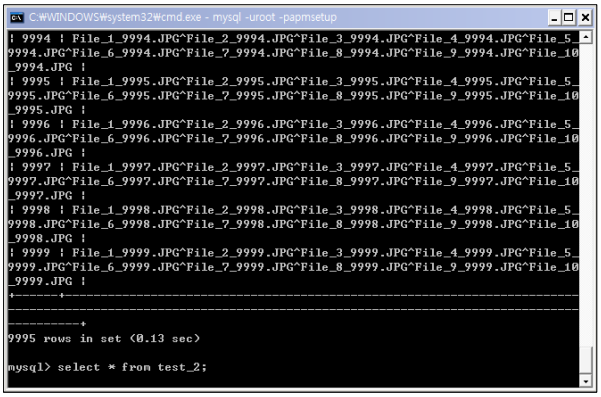
표 2. 검색 속도측정

SELET * FROM 구문을 이용하여 총 9995개의 레코드를 검색했을 때 걸린 시간

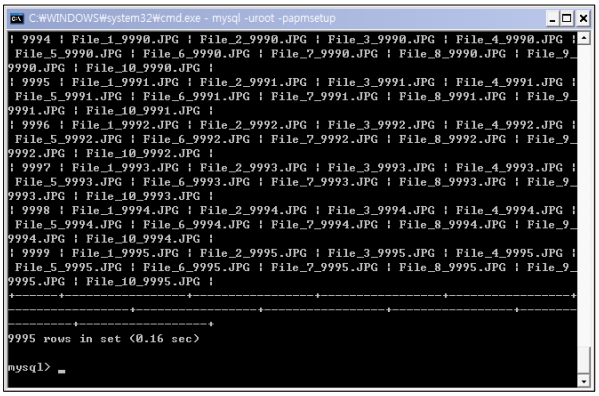
단일 DB 필드(Sec)	다중 DB 필드(Sec)	검색 속도 차
0.013	0.016	120%

표 3. 검색 속도측정

단일필드로 구성된 테이블을 SELET * FROM 구문으로 속도측정



다중필드로 구성된 테이블을 SELET * FROM 구문으로 속도측정



실험결과 다수 파일의 업로드 속도는 본 연구에서 제안한 단일 DB 필드방식이 기존의 다중 DB 필드방식에 비해 약 3배 이상 처리속도가 향상됨을 실험을 통해 확인할 수 있었고, 또한 파일 검색에서도 단일 DB 필드 방식이 기존의 방식에 비해 평균 2배 이상 처리속도가 향상된 결과를 보였다.

Ⅳ. 결론

인터넷 분야에서 파일 업로드 및 검색 등의 데이터 처리 속도향상에 대한 기술방식은 상당히 다양하게 발전되어져 오고 있으며 앞으로도 계속적으로 진보해야할 분야이다.

현재 웹에서 클라이언트의 파일을 서버로 저장하기 위해 업로드 시킬 때 파일은 해당 서버의 특정 폴더에 저장되고, 파일 이름만 필드에 저장하여 처리하는 방식이 활용되고 있다. 이를 위해서 DB는 업로드 할 파일의 개수만큼 필드를 생성하는 다중 필드방식이 사용되어져 오고 있다.

다중 필드 방식은 파일이 없는 경우에는 무효 필드를 가지며 예상한 고정 필드 경우보다 파일이 많은 경우에는 여러 번에 걸쳐 업로드가 실행되어져야하는 제한을 갖는다. 또한 수정, 삭제 등을 위한 검색 시에도 파일 명의 모든 필드를 검색해야하는 단점을 갖는다.

본 논문에서 제안한 단일 필드를 이용하는 방식을 PHP를 이용하여 구현하였으며, 이를 모의실험 한 결과, 기존의 다중 필드를 이용한 업로드 방식에 비해 파일 업로드 시 평균 약 3배 이상의 속도 빨라짐과 DB 검색 시에도 평균 2배 이상의 처리 속도가 향상됨을 확인할 수 있었다. 또한 업로드 파일의 개수를 예측할 수 없는 유동적인 파일 업로드 시에도 제안한 단일 필드 방식의 유용성을 확인함으로써 파일 업로드 방식을 활용하는데 널리 활용될 수 있음을 보였다.

앞으로 ASP, JSP 등과 같은 다른 웹 프로그래밍 프로그램에 대해서도 단일 필드 방식의 유효성을 검증할 계획이다.

참고문헌

[1] 강구홍, 황헌주, “RFC 1867 규격을 준수하는 ASP 업로드 컴포넌트 설계” 한국콘텐츠학회논문지, Vol. 6 No. 3 2006. 2

[2] 장계선, 가상 네트워크 스토리지 설계 및 구현 2007. 8

[3] 홍정순, 대용량 학습 자료 공유를 위한 웹 스토리지 기반의 홈페이지 설계 및 구현 2006. 8

PHP File Upload implementation of infinite using a single Database Field

Ki Hwan Hwang, Ki Hwan Byun, Jie Young Lee

Abstract

Currently the Web, multiple files in the Client to upload to the server of the multi-upload method, files are stored in specific folders on the server, the file name field in DB(Database Field) stored in a way that the process has been widely utilized. Therefore, a minimum number of file upload as many file names in the Database and the Table should be created in the field, but also a variable number of files can be uploaded has no limits. In this paper we proposed the new method that files to be uploaded to the server, you can upload a limited number of files, nor due to upload files to reduce load on the database to improve the processing speed of a single field using a multi-method file uploads using PHP directly implemented