

Studijní materiály předmětu Technologie WWW

Michaela Plisková

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická
Institut řízení procesů a aplikované informatiky

Vložit oficiální zadání bakalářské práce

Děkuji panu Ing. Tomášovi Dulíkovi za odborné vedení práce a za cenné připomínky, které mi v průběhu vypracování práce poskytl.

ABSTRAKT

Abstrakt česky

Úkolem této bakalářské práce bylo vytvořit studijní materiál pro předmět Technologie WWW. Tento předmět je zaměřen na tvorbu dynamických webových stránek – PHP. PHP spolupracuje také s databázemi, proto je část práce věnována databázi MySQL. Práce je rozdělena na 2 části. V první je popsána teorie PHP a MySQL, ve druhé části jsou vytvořeny praktické příklady a závěrečný ukázkový systém s udržováním kontextu.

Abstrakt ve světovém jazyce

The main purpose of my diploma was to create the study material for a subject Technology of WWW. This subject is specialized to production of dynamical web sites - PHP. PHP is in cooperation with databases, which is on of the part of this diploma apply to the database MySQL. This project is divided into two parts. In the first part of this diploma is described a theory of PHP and MySQL, in the second part are created schedules and a final specimen system with maintenance of context.

OBSAH

ÚVOD.....	7
I. TEORETICKÁ ČÁST.....	8
1 ZÁKLADY JAZYKA PHP	9
1.1 Instalace.....	9
1.2 Syntaxe.....	10
1.3 Proměnné.....	11
1.4 Příkazy	12
2 FORMULÁŘE	16
3 FUNKCE.....	19
3.1 Globální proměnné	20
4 OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ A PHP.....	21
4.1 Třídy	21
5 KNIHOVNÍ FUNKCE.....	23
5.1 Funkce pro práci se soubory.....	23
5.2 Funkce filesystému.....	24
5.3 Funkce pro práci s řetězci.....	26
5.4 Funkce pro práci s poli.....	27
6 COOKIES.....	29
7 SESSIONS	30
8 DATABÁZE	31
8.1 Relace.....	31
8.1.1 Relace 1 : 1	31
8.1.2 Relace N : 1	32
8.1.3 Relace N : M.....	33
8.2 Primary Key.....	33
8.3 SQL	33
8.4 Souvislost s PHP.....	34
9 MYSQL.....	35
9.1 Stavové proměnné.....	35
9.2 Funkce MySQL.....	37
9.3 Datové typy.....	39
9.4 Modifikátory	40
9.5 Jednoduchý řádkový klient mysql	41
9.6 Základní práce s MySQL	41
9.7 Dotazy	44
9.8 Komunikace PHP s MySQL	46
II. PRAKTICKÁ ČÁST.....	47
10 ÚKOL 1.....	48
11 ÚKOL 2.....	49
12 ÚKOL 3.....	51
13 ÚKOL 4.....	53
14 ÚKOL 5.....	54
15 SYSTÉM S UDRŽOVÁNÍM KONTEXTU.....	55

ZÁVĚR.....	59
SEZNAM POUŽITÉ LITERATURY	60
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	61
SEZNAM OBRÁZKŮ.....	62
SEZNAM TABULEK	63
SEZNAM PŘÍLOH.....	64

ÚVOD

Cílem mé bakalářské práce bylo vytvoření studijních materiálů pro předmět Technologie WWW. Tento předmět se zabývá vytvářením webových aplikací, se zaměřením na dynamické webové stránky – PHP.

Hlavní část je vypracována formou přednášek v programu MS Power Point. Příklady a podrobnější vysvětlení některých částí, patřící k jednotlivým přednáškám, jsem vypracovala v textovém editoru MS Word. Skripty PHP jsou vytvořeny pomocí Poznámkového bloku.

Teoretická část je věnována problematice PHP. Rozebírají se zde základy PHP, formuláře, funkce, knihovní funkce, sessions, cookies. Protože PHP úzce spolupracuje s databázemi, je část věnována také databázím. Obsahuje teorii k databázím SQL a rozebírá základy databáze MySQL – funkce, stavové proměnné, ..., komunikaci PHP s MySQL.

V praktické části je uvedeno několik jednoduchých úkolů, které by měl student vypracovat na hodině. Týkají se témat probraných na hodinách. Na ukázkou je vytvořen jednoduchý systém s udržováním kontextu.

I. TEORETICKÁ ČÁST

1 ZÁKLADY JAZYKA PHP

PHP je programovací jazyk sloužící k vytváření dynamických webových stránek. Vytvořil ho v roce 1994 Rasmus Lerdor. Je to skriptovací jazyk, který se provádí na straně serveru. Na tomto severu je závislý, protože na něm běží jeho interpreter, který skripty provádí. Vytvořené PHP skripty se zapisují přímo do HTML stránky (nejčastěji s příponou *.php). Výhodou tohoto jazyka je, že ke zdrojovým kódům skriptů se nedostane nikdo jiný než autor, nevýhodou, že nemůže (například oproti JavaScript) dynamicky reagovat na událost způsobenou klientem - k provedení každé své nové události musí být vždy prohlížečem znovu odeslán požadavek na server. Proto je velmi výhodné kombinovat PHP s dynamicky reagujícím jazykem. Důležitou součástí PHP je také spolupráce s databázemi. Podporuje MySQL, PostgreSQL, MS SQL server, ... S databázemi může pracovat i pomocí rozhraní ODBC (Open DataBase Connectivity) – čerpání dat např. z MS Access nebo z MS Excelu.

Jazyk obsahuje knihovny některých internetových protokolů, jako je HTTP, FTP, POP3, SMTP a jiné. Může se používat ve většině operačních systémů (Windows, Linux, ...) a je zcela zdarma. Největším konkurentem je technologie ASP od Microsoftu, ta se však platí a funguje správně pouze na počítači s operačním systémem Windows [1], [2] .

1.1 Instalace

Aby bylo možné PHP používat, musí být na počítači nainstalovaný nějaký interpreter [3]. PHP běží pouze na webserveru, se kterým komunikuje přes CGI rozhraní. Proto je třeba použít nějaký webserver, který tohle rozhraní CGI podporuje.

Existuje tzv. loopback systém - přesměrování pomocí URL (Uniform Resource Locator = adresa dokumentu) na lokální počítač, na kterém běží webserver, který spustil interpreter.

Tento systém nevyžaduje připojení k internetové síti, ale musí mít nainstalován Telefonní adaptér a TCP/IP protokol, přes který se připojí na lokální webserver. Mezi nejznámější webservery patří Apache [4].

HTML stránky obsahující PHP skript musí být volány přes svou URL adresu, která je `http://localhost/` + adresář na serveru Apache, který obsahuje stránky s PHP skripty + jméno skriptu.

Freeware editory PHP skriptů

- Maguma Studio
- EasyPad

1.2 Syntaxe

Vkládání PHP do HTML

Realizuje se pomocí speciální syntaxe, dané jazykem PHP. Začátek skriptu se označuje `<?` nebo `<?php` a konec se značí pomocí `?>`. Vše, co se vloží mezi tyto značky, považuje server za kód jazyka PHP. Může se použít i zápis, který se podobá vkládání JavaScriptů do HTML stránek. Začíná `<SCRIPT LANGUAGE="php">` a končí `</script>`.

Poznámky

Přímo do skriptů se mohou psát poznámky. Za poznámku je považován text mezi `//` a koncem aktuálního řádku. Pro delší poznámky se používají znaky `/*` a `*/`, mezi které se poznámka vepíše.

Oddělování příkazů

Při vkládání více příkazů, musí být od sebe jednotlivé příkazy odděleny. Mohou být od sebe odděleny středníkem nebo je každý příkaz uveden samostatně mezi TAGy.

První způsob:

```
<? echo "Pomocí"; echo "středníků."; ?>
```

Druhý způsob:

```
<? echo "Pomocí" ?><? echo "TAGů" ?>
```

Malá a velká písmena

Při deklarování proměnných se musí rozlišovat malá a velká písmena. Oproti tomu při psaní příkazů se mohou používat libovolně.

1.3 Proměnné

Jsou to prostředky sloužící k uchování hodnot v rámci skriptu. Může v nich být uloženo cokoli – typ hodnoty si určí PHP samostatně. Nemusí se deklarovat předem. Název proměnné začíná symbolem \$, poté následuje libovolná kombinace písmen a číslic. První znak za symbolem \$ však nesmí být číslo. Hodnota se přiřazuje pomocí znaku =.

Datové typy

- Integer - celé číslo v rozsahu od -2 147 483 648 do 2 147 483 647
- Double - desetinné číslo v rozsahu od $-1,7 \times 10^{308}$ do $1,7 \times 10^{308}$
- String – hodnotou je znakový řetězec. Zapisuje se mezi uvozovky nebo apostrofy. Pokud je potřeba uvést uvozovky nebo apostrofy přímo v řetězci, používají se tzv. escape sekvence.

Tab. 1 Únikové sekvence

Escape sekvence	Výsledek v řetězci
\\	zpětné lomítko
\“	uvozovky
\\$	znak dolaru
\n	začátek nového řádku
\t	tabulátor
\r	návrat na začátek řádku

- Array (pole) – proměnná, která obsahuje více hodnot. Každá hodnota proměnné se vyvolává pomocí indexu prvku pole. Tyto indexy se zapisují do hranatých závorek za název proměnné.

a) Index přidělí PHP samostatně:

```
$mesta [] = "Brno";  
$mesta [] = "Olomouc";  
$mesta [] = "Blansko";
```

b) Index se přidělí ručně:

```
$staty [5] = "CR";  
$staty[8] = "De";  
$staty [3] = "sk";
```

c) Použije se funkce array():

```
$mesta = Array("Brno", "Olomouc", "Blansko");
```

- Object – hodnotou je objekt [1], [8]

1.4 Příkazy

If

Příkaz `if` je jedním z nejdůležitějších příkazů každého programovacího jazyka. Slouží k podmíněnému provedení příkazu, pokud je splněna určitá podmínka. Ta se musí zadat jako výraz, který vrací logickou hodnotu.

```
if ($y!=0)
```

```
$vysledek=$x/$y
```

If-else

Tento příkaz je rozšířením příkazu If. Příkaz uvedený za else se provede v případě nesplnění dané podmínky.

```
if ($ x > $ y)
    echo „Správně“;
else    echo „Špatně“;
```

If-elseif-else

Modifikace předchozích příkazů. Tento příkaz se použije, pokud se má provést příkaz při nesplnění podmínky, ale zároveň při splnění jiné podmínky.

```
if ($ x < $ y)
    echo „x je menší než y“;
elseif ($ x == $ y) echo „x rovná se y“;
else echo „x je větší než y“;
```

Switch

Na základě hodnoty jednoho výrazu se provede jedna z několika větví skriptu. Nejprve je vyhodnocen výraz (proměnná) za příkazem switch, poté jsou postupně procházeny hodnoty uvedené za klíčovým slovem case, dokud se nenalezne hodnota shodná s hodnotou výrazu. Následně jsou vykonávány příkazy, dokud se nenarazí na break nebo konec příkazu switch. Pokud není nalezena žádná odpovídající hodnota, jsou provedeny příkazy za default. Default není povinné, ale většinou se používá.

```
switch ($stat) {  
    case "cz":  
        echo "Česká republika";  
        break;  
    case "sk":  
        echo "Slovenská republika";  
        break;  
    default:  
        echo "Odjinud";}
```

While

Vyhodnocuje logický výraz. Pokud výraz není pravdivý, přeskočí kód uzavřený uvnitř závorek, pokud je pravdivý, kód v závorkách se provádí. Program dojde ke konci a pokud je podmínka opakování vyhodnocena jako pravda, je kód uvnitř závorek proveden znovu. To se opakuje až do té doby, kdy je podmínka vyhodnocena jako nepravdivá. Kód může být uprostřed běhu přerušen pomocí příkazu break.

```
$i = 0;  
while($i < 10)  
{    echo "$i";  
    $i++; }
```

Do-while

Obdobný jako while, avšak podmínka je až na konci cyklu. Příkazy v tomto cyklu jsou vykonány alespoň jednou, oproti cyklus while, kde nemusí být provedeny ani jednou.

```
$i = 0;  
do  
{ echo "$i";  $i++;  
} while ($i < 10);
```

For

Do závorek se zadávají tři výrazy oddělené středníky. První je příkaz přiřazení pro inicializaci řídicí proměnné cyklu. Toto přiřazení se provede jen jednou. Druhý je logický výraz, který se vyhodnocuje na začátku každé iterace. Pokud je vyhodnocen jako pravda, iterace se provede, v opačném případě cyklus skončí. Třetí je příkaz, který je proveden na konci každé iterace cyklu.

```
for ($i = 1; $i <= 10; ++$i)  
echo („$i“);
```

Break

Okamžitě ukončí provádění cyklu.

Continue

Okamžitě přeskočí příkazy ve zbývajících částech těla cyklu a začne provádět další iteraci.

Require

Slouží k načtení skriptu ze souboru.

Include

Příkaz podobný jako require, rozdíl je v tom, že jeden příkaz se může volat několikrát. Například uvnitř cyklu [1], [6], [8].

2 FORMULÁŘE

Pro vytvoření interaktivní aplikace na webu jsou třeba formuláře, formulářové prvky. Tvorba formulářů se v jazyce PHP dělí na 2 části. Nejdříve se vytvoří samostatný formulář HTML a potom se vytvoří vhodný skript v jazyce PHP, který přijme uživatelem zadaná data.

Formulář HTML je tvořen dvěma značkami prvku `<FORM>` a několika prvky `<INPUT>`. V prvku `<FORM>` se nastavuje atribut `ACTION="navez_skriptu.php"`, který určuje jméno skriptu, kterému budou data z formuláře předána a atribut `METHOD="POST nebo GET"`. Tento atribut určuje, jak budou data z formuláře odeslána. Metodou GET budou hodnoty formuláře předány skriptu za URL adresou. Používá se pro přenášení menších hodnot. Metoda POST se používá pro přenášení větších dat v těle HTTP požadavku (skrytě). Obě metody jsou funkčně stejné, liší se jen přenosem dat.

Vstupní pole

```
<INPUT TYPE="TEXT" NAME="...">
```

Formulářový prvek, který se používá k činnostem, kdy se mají zjistit údaje od uživatele a odeslat je zpět skriptu. Atribut NAME určuje, jak se bude jmenovat proměnná, kterou skriptu odešleme. V této proměnné bude uložen obsah tohoto pole.

Mezi TYPE a NAME se mohou použít další atributy VALUE, MAXLENGTH a SIZE. Atribut VALUE určuje počáteční hodnotu, která bude v tomto vstupním poli již předepsána. Atribut MAXLENGTH určuje, kolik znaků se může do tohoto vstupního pole napsat. Atribut SIZE určuje velikost vstupního pole.

Odesílání formuláře

```
<INPUT TYPE="SUBMIT" VALUE="...">
```

Vyplněné hodnoty předá skriptu, který je zpracuje. Atribut TYPE="SUBMIT" stanovuje, že se jedná o odesílací tlačítko. Atribut VALUE="..." určuje text, který bude na tlačítku umístěn.

Vstupní pole – heslo

```
<INPUT TYPE="PASSWORD" VALUE="...">
```

Toto pole je určeno k zadávání hodnot od uživatele, které nemají být viditelné. Používá se většinou pro heslo. Místo znaků se zobrazují hvězdičky.

Přepínací pole – tlačítka

```
<INPUT TYPE="RADIO" VALUE="...">
```

V případě, kdy má uživatel na výběr z několika možností, se používá tento formulářový prvek. Může to být výběr například pohlaví, věková kategorie. Může se nastavit atribut CHECKED, který znamená, že tento prvek již bude vybrán.

Zaškrťovací pole

```
<INPUT TYPE="CHECKBOX" NAME="...">
```

Tento prvek se používá, pokud má uživatel na výběr několik možností. Je stejný jako prvek přepínací pole, liší se pouze v zobrazení.

Pole reset

```
<INPUT TYPE="RESET" VALUE="...">
```

Používá se v případě, kdy se má vymazat hodnota formuláře, formulářových elementů. Atributem VALUE se nastaví text, který bude vepsán v tomto tlačítku.

Skryté pole

```
<INPUT TYPE="HIDDEN" NAME="..." VALUE="...">
```

Tento element se používá například pro předání proměnných z jednoho skriptu do druhého.

Příklad:

```
<FORM ACTION="zadani.php" METHOD="post">  
<INPUT TYPE="HIDDEN" NAME="prvni" VALUE="30">  
<INPUT TYPE="HIDDEN" NAME="druhy" VALUE="60">  
<INPUT TYPE="SUBMIT" VALUE="Odeslat">  
</FORM>
```

Skript, který data zpracuje:

```
<?php  
echo "Hodnota proměnné $prvni je $prvni";  
echo "Hodnota proměnné $druhy je $druhy";  
?>
```

Seznam

```
<SELECT NAME=".."> </SELECT>
```

Tento formulářový prvek se používá v případě, kdy má uživatel na výběr několik možností. Atribut MULTIPLE určuje, že si uživatel může vybrat více možností. Atribut SIZE určuje počet najednou zobrazených řádků tohoto seznamu. Položky se do seznamu přidávají pomocí <OPTION VALUE="..."> VALUE. Hodnota uvedená za znakem > znamená hodnotu, která bude zobrazena v seznamu - nemá žádnou vazbu na atribut VALUE="...".

Víceřádkové textové pole

```
<TEXTAREA> </TEXTAREA>
```

Tento element umožňuje zapsání několika řádkového textu. Pomocí atributu COLS se nastavuje počet sloupců (šířku tohoto elementu) a atribut ROWS určuje počet řádků [8].

3 FUNKCE

V podstatě jsou to malé části skriptů, které je možné opakovaně používat. Do funkce může být vložen jakýkoli platný PHP kód, v PHP 3 musí být funkce definovány dříve, než je na ně odkazováno, v PHP 4 už tento požadavek neplatí. Každá funkce začíná klíčovým slovem function. Za slovem function je název funkce [1], [2], [6], [12].

```
function vypocet()
{
    $x = 10;
    $y = 20;
}

$x = 100;
$y = 200;
vypocet();
$vysledek = $x + $y;
echo $vysledek;
```

Funkce bez argumentu

Nemá uvnitř závorek žádný argument (proměnnou):

```
function vypis()
{ echo("ahoj"); }
```

Funkce s argumentem

```
function vypis ($jmeno){ echo($jmeno); }

vypis(Ahoj);

vypis (Dobrý den);
```

Funkce vracející hodnotu

```
function vrat($hodnota)
{
    return $hodnota*2;
}

echo(vrat(20)+"<br>");

echo(vrat(100)/vrat(10));
```

3.1 Globální proměnné

Pokud jsou ve skriptech definovány nějaké vlastní funkce, mohou se ve skriptu vyskytovat dvě proměnné se stejným identifikátorem (jménem). Proměnné, které jsou definovány uvnitř funkce, nemají žádný vliv na proměnné umístěné mimo definici funkce, a to i v případě, že mají stejný identifikátor. Proměnným, které jsou definovány mimo tělo funkce, se říká globální proměnné. K přiřazení globální proměnné slouží příkaz `global`. Za ním následuje čárkami oddělený seznam globálních proměnných, které mohou být ve funkci použity.

```
function vypocet()
{
    global $x;
    $x = 10;
    $y = 20;
}

$x = 100;
$y = 200;
vypocet();
$vysledek = $x + $y;
echo $vysledek;
```

4 OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ A PHP

4.1 Třídy

OOP umožňuje definovat tzv. třídy, což jsou speciální datové typy, které mohou obsahovat proměnné (data), ale zároveň i funkce pracující s těmito daty. Do jedné třídy se mohou umístit všechny proměnné a funkce obsluhující nějakou událost. Tím se dosáhne zpřehlednění celého skriptu, protože se nemusí definovat několik funkcí a globálních proměnných, ale nadefinuje se pouze třída, která bude danou událost obsluhovat.

Pokud je nadefinována nějaká třída, mohou se ve skriptu vytvářet její instance - objekty. Objekt je proměnná typu třída, pomocí níž se přistupuje k datům a funkcím v tomto objektu. Definice třídy:

```
class jméno třídy
{
    definice členských proměnných;
    definice členských funkcí;
}
```

Členské proměnné

Definice členské proměnné se ve třídách uvozuje klíčovým slovem `var`, za kterým je čárkami oddělený seznam proměnných.

Pokud je vytvořena třída, mohou se vytvářet její instance – objekty. Ty se tvoří pomocí klíčového slova `new`. K jednotlivým členským proměnným třídy se přistupuje pomocí operátoru `->`.

Členské funkce

Členské funkce se většinou definují pro manipulaci s daty uvnitř objektu. Speciální proměnná `$this->členská proměnná` se používá pro přístup ke členské proměnné uvnitř

členské funkce. Obsah této proměnné je při zavolání funkce nahrazen objektem, ze kterého je funkce volána.

Konstruktory

Konstruktor představuje speciální členskou funkci, která má shodný název s názvem třídy. Konstruktor je zavolán při každém vytváření objektu pomocí operátoru `new`. Nejčastěji se využívá pro snadnější inicializaci objektu.

Dědičnost

Dědičnost je možnost vytvořit novou třídu, která zdědí všechny členské proměnné a členské funkce z nějaké jiné třídy. V nově vytvořené třídě pak mohou být definované další členské proměnné a funkce [1], [7].

5 KNIHOVNÍ FUNKCE

V následující kapitole jsou uvedeny důležité knihovní funkce jazyka PHP. Bohužel není v možnostech této práce uvádět u každé funkce kompletní popis všech parametrů a návratové hodnoty – pro tyto informace si vás dovoluji odkázat na manuál PHP [6].

5.1 Funkce pro práci se soubory

- **fOpen** – funkce otevře soubor nebo URL-adresu
`$fp=fopen(„název souboru“, „mód“, „použití include_path“);`
`$fp=fopen("kontakt.php","a+");`
pokud se jako hodnota posledního nepovinného parametru `<použití include_path>` uvede `true`, bude soubor pro otevření hledat i v adresářích uvedených v direktivě `include_path` v konfiguračním souboru `php.ini`, jestliže se soubor nepodaří otevřít, vrací funkce `false`; parametr „mód“ určuje, v jakém režimu se má soubor otevřít
- **fClose** – funkce zavře soubor otevřený funkcí `fOpen()` nebo `FsockOpen()`
pokud se soubor podaří zavřít, vrací hodnotu `true`, v opačném případě `false`
- **fGets** – standardně načte jeden řádek ze souboru, musí se omezit maximální délka tohoto řádku hodnotou druhého parametru, standardně 1 kB
- **fRead** – čte z datového souboru po blocích,
`$cteni=fread($fp2,1024);` -- načte 1 Kb
- **fWrite** – pro zápis do souboru, prvním parametrem je odkaz na otevřený soubor, do něhož se mají data zapsat, druhým parametrem je řetězec, který se má do souboru zapsat
`fwrite(soubor,řetězec)`
- **fTell** – vrací aktuální pozici ukazatele v souboru
- **rewind** – nastaví ukazatel pozice na počátek zadaného souboru
- **fSeek** – nastaví pozici v daném souboru na určenou hodnotu
- **fGetc** – načte ze vstupního souboru jeden znak
- **fPuts** – запиše řetězec do souboru, je synonymem k funkci `fWrite()`

5.2 Funkce filesystemu

- `basename` - vrátí část cesty obsahující název souboru
- `chgrp` - změni skupinu souboru
- `chmod` - změni mód souboru
- `chown` - změni vlastníka souboru
- `clearstatcache` - vymaže cache stavu souborů
- `copy` - zkopíruje soubor
- `dirname` - vrací část cesty obsahující název adresáře
- `diskfree` - vrátí diskový prostor dostupný v adresáři
- `fclose` - zavře otevřený deskriptor souboru
- `feof` - test na konec souboru
- `fflush` - zapíše obsah výstupního bufferu
- `fgetc` - načte 1 znak ze souboru
- `fgets` - přečte řádek ze souboru
- `fgetss` - přečte řádek ze souboru a odstraní HTML značky
- `file_exists` - zjistí, zda soubor existuje
- `file` - načte celý soubor do pole
- `fileatime` - vrací čas posledního přístupu k souboru
- `filegroup` - vrací skupinu pro soubor
- `filemtime` - vrací čas změny souboru
- `fileowner` - vrací vlastníka souboru
- `fileperms` - vrací přístupová práva k souboru
- `filesize` - vrací velikost souboru
- `filetype` - vrací typ souboru
- `flock` - jednotné zamykání souboru
- `fpasssthru` - vypíše všechna zbývajících data v souboru

- `fscanf` - parsuje vstup ze souboru podle formátu
- `fseek` - posouvá ukazatel v souboru
- `fstat` - vrací informace o otevřeném souboru
- `ftell` - vrací pozici v souboru
- `ftruncate` - zkrátí soubor na danou délku
- `fwrite` - binárně bezpečný zápis do souboru
- `is_dir` - zjistí, zda je daný soubor adresářem
- `is_executable` - zjistí, zda je soubor spustitelný
- `is_file` - zjistí, zda se jedná o obyčejný (regular) soubor
- `is_link` - zjistí, zda se jedná o symbolický odkaz (link)
- `is_readable` - zjistí, zda lze ze souboru číst
- `is_uploaded_file` - zjistí, zda byl soubor uploadován pomocí HTTP POST
- `is_writable` - zjistí, zda lze do souboru zapisovat
- `link` - vytvoří hard link
- `linkinfo` - vrací informaci o odkazu (linku)
- `lstat` - zjišťuje informace o souboru nebo symbolickém odkazu
- `mkdir` - vytvoří adresář
- `set_file_buffer` - nastaví buffer pro soubor
- `stat` - zjišťuje informace o souboru
- `symlink` - vytvoří symbolický odkaz
- `tempnam` - vytvoří soubor s unikátním názvem
- `tmpfile` - vytvoří dočasný soubor
- `touch` - nastaví čas změny souboru
- `unlink` - smaže soubor

5.3 Funkce pro práci s řetězci

- `addslashes` - opatří řetězec lomítky ve stylu jazyka C
- `addslashes` - opatří řetězec lomítky
- `bin2hex` - převede binární data na hexadecimální reprezentaci
- `chop` - odstraní netisknutelné znaky z konce řetězce
- `chr` - vrátí určitý znak
- `chunk_split` - rozdělí řetězec na menší části
- `convert_cyr_string` - převede z jedné znakové sady Azbuky do jiné
- `count_chars` - vrátí informace o znacích použitých v řetězci
- `crc32` - spočítá kontrolní součet `crc32` řetězce
- `crypt` - jednosměrné zašifrování řetězce
- `echo` - vytisknutí jednoho nebo více řetězců
- `explode` - rozdělí řetězec jiným řetězcem
- `hebrew` - převede logický Hebrejský text na vizuální text
- `hebrevc` - převede logický Hebrejský text na vizuální text s konverzí konců řádků
- `htmlentities` - převede všechny použitelné znaky na HTML entity
- `htmlspecialchars` - převede zvláštní znaky na HTML entity
- `implode` - spojí prvky pole pomocí řetězce
- `join` - spojí prvky pole pomocí řetězce
- `levenshtein` - spočítá XXX Levenshteinovu vzdálenost mezi dvěma řetězci
- `ltrim` - odstraní netisknutelné znaky ze začátku řetězce
- `md5` - spočítá MD5 hash řetězce
- `metaphone` - spočítá metaphone klíč řetězce
- `nl2br` - před všechny konce řádků v řetězci vloží HTML konce řádků
- `ord` - vrátí ASCII hodnotu znaku
- `parse_str` - rozparsuje řetězec do proměnných

- `print` - vytiskne řetězec
- `printf` - vytiskne formátovaný řetězec
- `quoted_printable_decode` - převede quoted-printable řetězec na osmibitový řetězec

5.4 Funkce pro práci s poli

- `array_count_values` - spočítá všechny hodnoty v poli
- `array_diff` - spočítá rozdíl polí
- `array_flip` - prohodí klíče a hodnoty pole
- `array_intersect` - spočítá průnik polí
- `array_keys` - vrátí všechny klíče pole
- `array_merge_recursive` - rekurzivně sloučí dvě nebo více polí
- `array_merge` - sloučí dvě nebo více polí
- `array_multisort` - třídí více polí, nebo vícerozměrné pole
- `array_pad` - doplní pole hodnotou na určenou délku
- `array_pop` - odstraní prvek z konce pole
- `array_push` - přidá jeden nebo více prvků na konec pole
- `array_rand` - vybere náhodně jeden nebo více prvků pole
- `array_reverse` - vrátí pole s prvky v opačném pořadí
- `array_shift` - odstraní prvek ze začátku pole
- `array_slice` - vytáhne část pole
- `array_splice` - odstraní část pole a nahradí ji něčím jiným
- `array_unique` - odstraní z pole duplicitní hodnoty
- `array_unshift` - připojí jeden nebo více prvků na začátek pole
- `array_values` - vrátí všechny hodnoty v poli
- `array_walk` - použije uživatelskou funkci na všechny prvky pole

- array - vytvoří pole
 - arsort - třídí pole sestupně se zachováním klíčů
 - asort - třídí pole se zachováním indexů
 - compact - vytvoří pole obsahující proměnné a jejich hodnoty
 - count - spočítá prvky v proměnné
 - current - vrátí současný prvek pole
 - each - vrací další klíč/hodnota pár z pole
 - end - nastaví vnitřní ukazatel pole na jeho poslední prvek
 - in_array - ověří, zda v poli existuje daná hodnota
 - krsort - třídí pole sestupně podle klíčů
 - ksort - třídí pole podle klíčů
 - list - přiřadí hodnoty proměnným, jako kdyby byly polem
 - next - posune interní ukazatel pole
 - pos - získá současný prvek pole
 - range - vytvoří pole obsahující rozsah integerů
 - reset - nastaví interní ukazatel pole na jeho první prvek
 - rsort - třídí pole sestupně
 - shuffle - zamíchá pole
 - sizeof - zjistí počet prvků v poli
 - sort - třídí pole
 - uasort - třídí pole pomocí uživatelsky definované porovnávací funkce se zachováním klíčů
 - uksort - třídí pole podle klíčů pomocí uživatelsky definované porovnávací funkce
 - usort - třídí pole podle hodnot pomocí uživatelsky definované porovnávací funkce
- [6], [7]

6 COOKIES

Jsou mechanismem na ukládání dat v uživatelské prohlídce, na sledování nebo identifikaci vracejících se uživatelů. V podstatě je to několik bajtů dat, které si webový server uloží v prohlídce přímo u uživatele. Jsou součástí HTTP hlavičky. Všechny cookies přijaté od klienta se automaticky stávají PHP proměnnou. Obsahem cookies může být cokoli, pouze celková velikost je omezena na 4 KB.

Cookies se nastavují pomocí funkce `setcookie(...)`. Tato funkce zapisuje do hlaviček HTTP protokolu a musí se použít dříve, nežli se cokoli vypíše na výstup. Pokud se tak nestane, funkce selže a nic nevyšle. Pro příjem cookies je určeno pole proměnných s názvem `$HTTP_COOKIE_VARS`, které obsahuje všechna cookies, které webová stránka přijala.

Parametry:

- a) jméno – každá cookies musí mít své jméno
- b) data – ta, co jsou uložena do cookies
- c) doba expirace - představuje datum a čas, kdy ji prohlížeč automaticky odstraní; pokud se nezadá žádné datum expirace, automaticky se smaže při zavření stránky
- d) doména - cookies platí pro určitou doménu
- e) cesta - cookies smí přijmout jen stránky s určitou virtuální cestou; pokud se neuvede žádná cesta, jsou cookies přístupné jen webovým stránkám ze stejného adresáře, jako je stránka, která cookies vyslala
- f) secure - určuje, zda cookies se smí přijímat pouze pomocí zabezpečeného SSL kanálu, nebo zda může přijít i normálním nezabezpečeným kanálem

Jednotlivé parametry se mohou zprava vynechávat.

Pro uvedení doby expirace se používají dvě funkce:

- a) `time` – slouží ke stanovení za kolik sekund od této chvíle má cookie vypršet
- b) `mktime` – stanoví se přesně datum a čas, kdy má dojít k vypršení cookie

7 SESSIONS

Je mechanismus podobný cookies, který funguje přímo na serveru, bez odesílání dat sem a tam. Jakmile PHP obdrží příkaz k započetí session, zjistí nejprve, zda již session neběží - pokud ne, vytvoří ji, pokud ano, připojí se k ní. PHP přidělí session identifikátor a vyhradí si někde místo pro ukládání tzv. session-proměnných. Jakmile je session nastartována, je proměnná "promenna" od této chvíle součástí session a je tudíž přístupná pomocí prvku "promenna" globálního asociativního pole \$_SESSION. Další libovolný PHP skript na stejném serveru bude mít k této proměnné přístup, za předpokladu, že bude součástí session. Je možné mít zároveň i celou řadu proměnných, které nebudou součástí session a které při ukončení skriptu zmizí. Session musí být nastartována předtím, než je jakýkoli výstup odeslán prohlížeči. Data spojená s probíhající session jsou umístěna většinou v souboru na serveru, a to mimo kořenový adresář serveru.

Výhody:

- a) spolehlivost – mechanismus session proměnných není na cookies závislý, dokáže v případě potřeby využít i jiných prostředků
- b) bezpečnost – potenciálně citlivá data není třeba přenášet mezi uživatelem a serverem a riskovat jejich prozrazení
- c) anonymita – identifikace sezení je zcela anonymní a nedává žádnou informaci ani o uživateli, ani o datech a serveru uložených [1], [2], [7], [9]

8 DATABÁZE

Programy, vytvořené v jazyce PHP, sice mohou používat k ukládání větších objemů dat souborového systému serveru, ale pokud je potřeba ukládat a zpracovávat větší objem dat, popř. pracovat s daty distribuovaně (například pokud jsou data uložena na vzdáleném serveru nebo serverech), je vhodné použít profesionální řešení databáze.

Zpracováním a přístupem údajů v databázi bývá pověřen program DBMS (DataBase Management System) česky SŘBD. Veškeré zpracování, ukládání a získávání dat je prováděno DBMS programem, proto jakákoli aplikace, která databázi používá, vždy přistupuje přes tento DBMS program. Mezi DBMS patří řada programů v nejrozličnějších cenových kategoriích. Existují programy, které jsou v nízké cenové relaci, nebo zcela zdarma - představitelem těchto DBMS systémů jsou například MySQL, PostgreSQL, nebo Firebird.

Většina dnešních DBMS staví na tzv. relačním modelu dat. V tomto modelu jsou veškerá data uspořádána do databázových tabulek. Každá databázová tabulka zpravidla uspořádává údaje o určitém objektu, je uspořádána do řádků a sloupců [2].

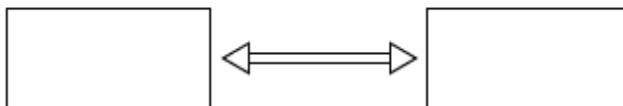
8.1 Relace

Mezi jednotlivými tabulkami mohou existovat tzv. relace, které mohou vytvořit vztahy mezi sloupci (poli). Může se tak propojit například tabulku učitelů s tabulkou tříd, která obsahuje seznam tříd.

8.1.1 Relace 1 : 1

Patří mezi nejzákladnější z typů relací a umožňuje vytvořit relaci mezi dvěma tabulkami. V praxi to znamená, že jeden učitel bude učit jednu třídu (řádek z jedné tabulky bude svázán s jedním řádkem z druhé tabulky).

Relace 1 : 1

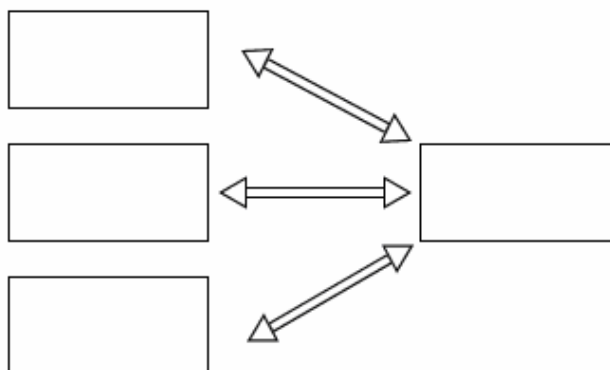


Obr. 1 Relace 1:1

8.1.2 Relace N : 1

Jedná se o relaci, kdy jeden řádek z jedné (primární) tabulky bude svázán s více řádky z tabulky sekundární. Příkladem může být například to, že jeden učitel může učit více tříd, ale každá třída může být současně vyučována pouze jedním učitelem.

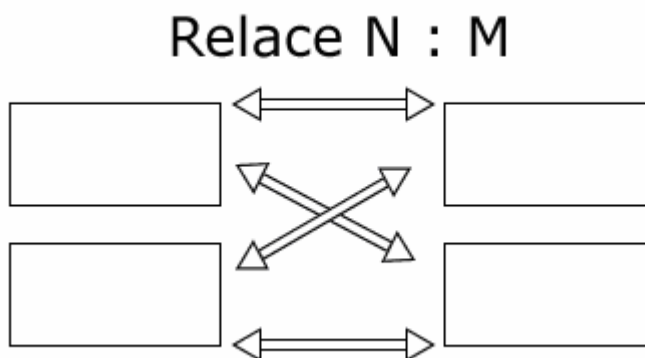
Relace N : 1



Obr. 2 Relace N:1

8.1.3 Relace N : M

Jedná se o nejkomplikovanější relaci, kdy několik řádků primární tabulky je svázáno s několika řádky sekundární tabulky. Jako příklad si můžeme uvést to, že jeden autor může napsat několik článků a jeden článek může být napsán více autory.



Obr. 3 Relace N:M

8.2 Primary Key

Pro jednoznačnou identifikaci každého řádku se užívají tzv. primární klíče. Primárním klíčem se může stát každý sloupec, který nebude mít v žádném řádku duplicitní hodnotu. Primárním klíčem se nesmí stát pole, které by mohlo obsahovat prázdnou hodnotu (NULL). Jako primární klíč se volí například rodné číslo (je pro každého občana unikátní) [13].

8.3 SQL

Aby mohly jiné aplikace používat údaje z databáze, musí DBMS program nabízet rozhraní, pomocí kterého budou moci s DBMS spolupracovat. Při práci s DBMS programem se používá osvědčený model klient/server. V roli serveru vystupuje DBMS program nabízející své služby. Je tak nepřetržitě spuštěn a čeká na požadavky od klientů - jiných aplikací. Na tyto požadavky pak odpovídá a reaguje na ně. Protože vystupuje v roli serveru, říká se mu také databázový server. Pro zápis požadavků na databázový server se dnes nejčastěji používá SQL jazyk .

SQL je anglická zkratka znamenající Structured Query Language. Tento jazyk dnes používá naprostá většina databázových serverů. Zjednodušeně se takový serverům říká SQL servery. Samotný SQL obsahuje vše potřebné pro práci s databázemi - ať už příkazy pro práci na vytvoření, rušení, modifikování tabulky, tak i pro práci s údaji samotnými - přidávání, změnu, rušení a vyhledávání údajů. Jazyk SQL má za sebou poměrně dlouhý vývoj. Jeho prototyp byl poprvé implementován v roce 1974 společností IBM. Standardem se stal v roce 1986, kdy se dočkal standardizace díky organizaci ANSI. Od té doby se tedy SQL stal univerzálním nástrojem pro programování databází. Jazyk SQL říká, co se má v databázi provést.

Skládá ze dvou základních částí, jednak z části DDL, a jednak z části DML.

Příkazy DDL jsou zkratkou za Data Definition Language. Slouží k definici a modifikaci tabulek a dalších databázových objektů.

Příkazy DML jsou zkratkou za Data Manipulation Language a slouží pro manipulaci s databázovými údaji.

8.4 Souvislost s PHP

V roli klienta pro databázový server může jako aplikace vystupovat i PHP. Může se spojit s databázovým serverem a využívat jeho služby, požadovat zápis, nebo získání určitých údajů z databáze. PHP skripty mohou obsahovat příkazy zapsané v jazyce SQL a PHP může zpracovávat výsledky SQL po provedení na databázovém serveru. V praxi má každý SQL server svůj vlastní protokol, a proto PHP musí umět s každým SQL serverem komunikovat trochu jinak. PHP umí komunikovat s mnoha SQL servery, mimo jiné třeba s Oracle, Sybase, Informix, MySQL, PostgreSQL.

9 MySQL

Je jednoduchý a nenáročný databázový systém. Etabloval se především ve webových aplikacích, je hodně preferovaný při spolupráci s PHP. Ve své podstatě je MySQL ořezaný o některé možnosti, které mají jiné databázové systémy. Důsledkem tohoto ořezání je nenáročnost MySQL na zdroje počítače a zvýšení rychlosti u některých operací [2].

9.1 Stavové proměnné

Hodnoty těchto proměnných obsahují aktuální stav MySQL serveru. Runtime proměnné nejsou konfigurační, takže je nelze měnit, jen sledovat.

- `aborted_clients` - počet násilně ukončených spojení, klient zmizel bez náležitého ukončení spojení
- `aborted_connects` - počet pokusů o připojení k serveru, které skončily nezdarem
- `bytes_received` - celkový počet bytů přijatý od všech klientů během všech spojení
- `bytes_sent` - celkový počet bytů odeslaných všem klientům během všech spojení
- `connections` - celkový počet pokusů o připojení k serveru.
- `created_tmp_disk_tables` - aktuální počet skrytých dočasných tabulek na disku serveru, které jsou vytvářeny během obsluhy požadavků
- `created_tmp_tables` - aktuální počet skrytých dočasných tabulek v operační paměti serveru, které jsou vytvářeny během obsluhy požadavků
- `created_tmp_files` - kolik dočasných souborů server vytvořil
- `handler_delete` - celkový počet všech smazaných řádků ve všech tabulkách a databázích, co server obsluhuje
- `handler_read_first` - celkový počet případů, kdy první údaj byl načítán z indexu tabulky; vztahuje se na všechny tabulky a databáze; pokud je hodnota vysoká, naznačuje to, že server provádí často prohledávání celých indexů

- handler_read_key - celkový počet požadavků čtení řádků, jejichž nějaký sloupec je v dané tabulce klíčem; pokud je hodnota vysoká, naznačuje to, že dotazy i tabulky jsou správně indexovány
- handler_read_next - celkový počet požadavků čtení řádků, jejichž nějaký sloupec je v dané tabulce v pořadí dalším (ne prvním) sloupcem s klíčem
- handler_read_rnd - celkový počet požadavků čtení řádků, kde se vyžaduje setřídění výstupní tabulky
- handler_read_rnd_next - celkový počet požadavků prohledávání dvojic řádků za sebou v pořadí tabulky
- handler_update - celkový počet všech kdy změněných řádků ve všech tabulkách a databázích, co server obsluhuje
- handler_write - celkový počet všech kdy zapsaných řádků ve všech tabulkách a databázích, co server obsluhuje
- key_blocks_used - počet aktuálně využitých bloků v cache klíčů
- key_read_requests - počet požadavků čtení bloku z cache klíčů
- key_write_requests - počet požadavků načtení bloku klíčů do cache klíčů
- key_writes - počet fyzických zápisů bloků klíčů na disk serveru
- max_used_connections - maximální počet současně používaných spojení
- not_flushed_key_blocks - počet bloků klíčů v cache klíčů, které byly změněny, ale nebyly dosud zapsány na disk
- not_flushed_delayed_rows - počet řádků čekajících na zapsání pomocí SQL příkazů INSERT DELAYED
- open_tables - celkový počet aktuálně otevřených tabulek
- open_files - celkový počet aktuálně otevřených souborů
- open_streams - celkový počet aktuálně otevřených proudů
- opened_tables - celkový počet kdy otevřených tabulek
- select_full_join - celkový počet operací spojení tabulek bez pomocí klíčů

- `select_full_range_join` - celkový počet operací spojení tabulek, kde server použil prohledávání v celém rozsahu připojované tabulky
- `select_range` - celkový počet operací spojení tabulek, kde server použil prohledávání v celém rozsahu první tabulky
- `select_scan` - celkový počet operací spojení tabulek, kde server procházel první tabulku
- `select_range_check` - celkový počet operací spojení tabulek bez použití klíčů, kde server kontroluje použití klíče v každém řádku tabulky
- `questions` - počet dotazů poslaných serveru
- `sort_merge_passes` - počet záměn, které se musely vykonat při setřídování
- `sort_range` - počet třídění, která byla uskutečněna v daném rozsahu
- `sort_rows` - počet všech tříděných řádků
- `sort_scan` - počet třídění, která byla uskutečněna pomocí procházení celé tabulky
- `table_locks_immediate` - aktuální počet zamknutých tabulek
- `table_locks_waited` - počet tabulek čekajících na uzamčení, např. aby na nich pak mohl být vykonán SQL dotaz
- `threads_connected` - počet aktuálně otevřených spojení
- `uptime` - kolik sekund již MySQL server běží [5]

9.2 Funkce MySQL

V následující kapitole jsou uvedeny některé funkce MySQL. Bohužel není v možnostech této práce uvádět u každé funkce kompletní popis všech parametrů – pro tyto informace si vás dovoluji odkázat na manuál MySQL [16].

- `mysql_db_query` - pošle MySQL dotaz
`$vypis = "select * from Zaznam";`
`$vysledek = mysql_db_query (,navez_databaze“, $vypis, $spojeni);`

- `mysql_affected_rows` - vrátí počet ovlivněných (změněných) záznamů v MySQL po posledním dotazu
- `mysql_change_user` - změni přihlášeného uživatele v současném spojení
- `mysql_client_encoding` - vrátí název znakové sady
- `mysql_close` - ukončí (zavře) MySQL spojení
- `mysql_connect` - vytvoří spojení s MySQL Serverem
- `mysql_create_db` - vytvoří MySQL databázi
- `mysql_data_seek` - přesune ukazatel na aktuální záznam
- `mysql_db_name` - vrátí seznam všech databází
- `mysql_drop_db` - vymaže (odstraní) MySQL databázi
- `mysql_errno` - vrátí číselnou hodnotu chybové hlášky předchozího MySQL příkazu
- `mysql_error` - vrátí text chybové zprávy předchozího MySQL příkazu
- `mysql_escape_string` - upraví řetězec pro bezpečné použití v `mysql_query`
- `mysql_fetch_array` - načte výsledný řádek do asociativního, číselného pole nebo do obojího
- `mysql_fetch_assoc` - načte výsledný řádek do asociativního pole
- `mysql_fetch_field` - načte informace o sloupci z výsledku do proměnné objektu
- `mysql_fetch_lengths` - zjistí délku všech položek aktuálního výstupu
- `mysql_field_type` - vrací typ specifikovaného sloupce
- `mysql_free_result` - uvolní výsledek z paměti
- `mysql_get_client_info` - načte verzi klientské knihovny MySQL
- `mysql_get_host_info` - načte informaci o hostu
- `mysql_get_proto_info` - načte informaci o protokolu MySQL
- `mysql_get_server_info` - načte informace o serveru MySQL
- `mysql_info` - vrací informace o posledním dotazu
- `mysql_list_dbs` - načte všechny dostupné databáze na MySQL serveru

- `mysql_list_fields` - načte výsledek s obsahem sloupce
- `mysql_list_tables` - načte všechny tabulky v MySQL databázi
- `mysql_num_fields` - vrací počet sloupců ve výsledku
- `mysql_num_rows` - vrací počet záznamů ve výsledku
- `mysql_pconnect` - otevře persistentní spojení s MySQL serverem
- `mysql_ping` - ověří spojení se serverem, případně, není-li spojení dostupné, pokusí se připojit znovu
- `mysql_query` - pošle MySQL dotaz
- `mysql_real_escape_string` - upraví řetězec pro bezpečné použití v `mysql_query`
- `mysql_result` - načte obsah jednoho sloupce tabulky
- `mysql_select_db` - nastaví MySQL databázi
- `mysql_stat` - vrací aktuální stav systému
- `mysql_tablename` - načte jméno tabulky

9.3 Datové typy

- `Int` – celé číslo celé číslo v rozsahu -2 147 483 648 až 2 147 483 647
- `Smallint` - celé číslo v rozsahu -32 768 až 32 767
- `Tinyint` - celé číslo v rozsahu 0 až 255
- `Float` - číslo s pohyblivou řádovou čárkou
- `Decimal (p)` - desetinné číslo s p platnými číslicemi
- `Decimal (p, d)` - desetinné číslo s p platnými číslicemi a d platnými desetinnými místy
- `Money` - číslo jako peněžní částka
- `Char (x)` - textový řetězec o délce x znaků (nejvíce však 255)
- `Varchar (x)` - textový řetězec o délce maximálně x znaků (nejvíce však 255)
- `Time` - čas ve formátu CQ:HH:SS

- Date - datum ve formátu RRRR-MM-DD
- Datetime - datum a čas ve formátu RRRR-MM-DD CQ:HH:SS
- Blob - speciální datový typ pro uložení binárních dat (soubory...)

9.4 Modifikátory

- AUTO_INCREMENT - systém si sám ve sloupci generuje unikátní (jedinečné) číselné hodnoty, lze ho použít pouze na celočíselný datový typ
- BINARY - pro CHAR a VARCHAR; tento typ bude brán jako binární a budou se tak rozlišovat malá a velká písmena
- DEFAULT vychozi_hodnota - pokud bude buňka prázdná, systém do ní automaticky přiřadí hodnotu "vychozi_hodnota"
- FULLTEXT INDEX - platí pro sloupce typu CHAR, VARCHAR a TEXT, fulltextový index slouží k rychlejšímu hledání dat v textových polích; hledání v takovýchto polích se provádí pomocí příkazů MATCH a AGAINST

př.: SELECT * FROM tabulka

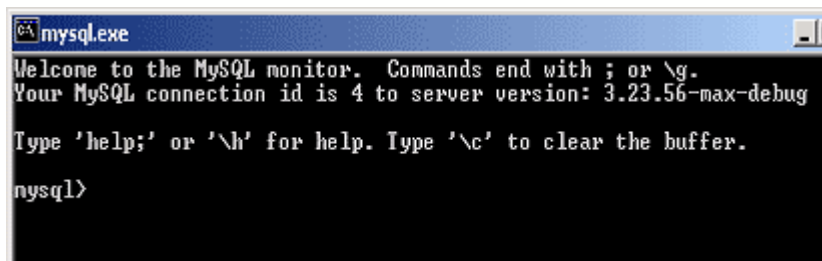
WHERE MATCH(sloupec) AGAINST("hledana_hodnota");

- NOT NULL - pokud se použije tento modifikátor, označený typ bude muset v každé buňce obsahovat nějakou hodnotu
- NULL - opak NOT NULL; buňka může být prázdná
- PRIMARY KEY - označený typ bude sloužit jako primární klíč - při jeho použití musí být zároveň použit UNIQUE - sloupec tedy jedinečným způsobem identifikuje záznamy v tabulce
- UNIQUE - v daném sloupci nesmějí být v buňkách stejné hodnoty
- UNSIGNED - datový typ bude bez znaménka a posune se interval hodnot, u čísel s pohyblivou desetinou čárkou se interval použitím UNSIGNED neposunuje a berou se jen kladná čísla
- ZEROFILL - použití u čísel, příkaz doplní před číslo nuly v celé jeho šířce [10]

9.5 Jednoduchý řádkový klient mysql

Součástí balíku MySQL je jednoduchý řádkový klient – mysql.exe.

Spouští se z : C:\.....\mysql\bin\mysql.exe.



Obr. 4 Řádkový klient MySQL

Toto okno se objeví pouze tehdy, jestliže je spuštěná i serverová část. Pak to znamená, že se podařilo spojit se serverovou částí a čeká se na zadání příkazů. Pokud není spuštěna serverová část, pak klient pouze pípne a ukončí se.

Základním příkazem je exit, který ukončuje práci s řádkovým klientem. Všechny příkazy se zobrazí po zadání příkazu help.

9.6 Základní práce s MySQL

Vytvoření nové databáze

Na začátku práce se musí vytvořit nová databáze. Samotná MySQL dokáže pracovat s mnoha databázemi, tedy úložišti dat, naráz. Každá databáze má své jméno. Názvy databází je dobré vybírat bez háčeků a čárek, protože ne každý MySQL server si s nimi poradí.

Vytvoření nové databáze Zkouska.

CREATE DATABASE Zkouska;

Tento SQL příkaz se zadá do řádkového klienta a potvrdí klávesou Enter.

Databáze Zkouska je vytvořena. Po spuštění řádkového klienta mysql je potřeba nastavit, s jakou databází má pracovat. Používá se příkaz: USE Zkouska;

Tímto příkazem se říká řádkovému klientovi, že od teď má používat databázi s názvem Zkouska.

Vytvoření databázové tabulky

Aby se mohli ukládat data do databáze, musí být vytvořena alespoň jedna databázová tabulka.

```
CREATE TABLE Data(  
Jmeno varchar(30),  
Příjmení varchar(35),  
RodneCislo varchar(11) NOT NULL,  
PRIMARY KEY (RodneCislo));
```

Nejpoužívanější typy sloupce jsou int pro celé číslo a varchar(n) pro textový řetězec o maximální délce n znaků.

Rodné číslo je použito jako tzv. primární klíč tabulky.

Přidání řádku do tabulky

K tomu slouží SQL příkaz INSERT:

```
INSERT INTO Data VALUES( 'Petr', 'Musil', '801023/2344');
```

Prohlížení zadaných řádků

Pro získávání už zadaných dat z databázových tabulek slouží SQL příkaz SELECT. Má mnoho podob a tváří, ale pro ten nejjednodušší případ, kdy se zobrazí všechna data v tabulce, má tvar:

```
SELECT * FROM Data;
```

Vybrání databáze

Aby se mohlo pracovat s určitou databází, musí být databáze vybrána. Každý databázový server může spravovat mnoho databází, proto je dobré hned po připojení databázi vybrat. K tomu slouží funkce:

```
mysql_select_db ( název_databáze, identifikátor_spojení )
```

Poslání SQL příkazu

K zaslání SQL příkazu se používá funkce `mysql_query`:

```
mysql_query ( sql_příkaz, identifikátor_spojení )
```

Funkce `mysql_query` vrací buď identifikátor výsledku, pokud SQL příkaz vrací nějaká data, nebo pouze hodnotu `true`, pokud SQL dotaz nevrací žádná data. Při chybě vrací funkce `mysql_query` vždy hodnotu `false`.

Čtení výsledků SQL dotazu

Pro čtení výsledků SQL dotazu existují dvě funkce.

a) `mysql_fetch_row ( identifikátor_výsledku )`

```
$vysledek = MySQL_DB_Query("php", $sql, $spojeni);
```

```
while($zaznam = MySQL_Fetch_Row($vysledek)):
```

```
echo $zaznam[0]."<br>";
```

```
echo $zaznam[1]."<br>";
```

```
echo $zaznam[2]."<br>";
```

```
endwhile;
```

```
b) mysql_fetch_array ( identifikátor_výsledku )

$vysedek = MySQL_DB_Query("php", $sql, $spojeni);

while($zaznam = MySQL_Fetch_Array($vysedek)):

    echo $zaznam[rc]."<br>";

    echo $zaznam[jmeno]."<br>";

    echo $zaznam[adresa]."<br>";

endwhile;
```

Obě funkce načtou další řádek z výsledků SQL dotazu a vrátí buď pole s hodnotami, a nebo false, pokud bylo už všechno přečteno. Liší se od sebe jen tím, že `mysql_fetch_row` načte hodnoty do pole s číselnými indexy, zatímco funkce `mysql_fetch_array` přidá i indexy s názvy sloupců.

9.7 Dotazy

Pro získávání jakýchkoliv informací slouží jediný příkaz `SELECT`, který je ze všech příkazů jazyka SQL nejsložitější, má nejbohatší strukturu a nejvíce možností .

Kompletní syntaxe vypadá následovně:

```
'select' :=
SELECT seznam sloupců
FROM seznam tabulek
[WHERE restrikce]
[GROUP BY výrazy pro seskupení]
[HAVING doplňující podmínky pro skupinu]
[UNION 'select']
[INTERSECT 'select']
[MINUS 'select']
[ORDER BY dle čeho třídít]
```

Restrikce

Omezování řádků pomocí podmínek

WHERE restrikce

Podmínka se skládá z kombinace logických výrazů a logických spojek AND, OR a NOT a kulatých závorek. Logické výrazy se mohou sestavovat z názvů sloupců, z relačních operátorů a speciálních klíčových slov. V jednotlivých částech logického výrazu mohou vystupovat aritmetické výrazy se základními matematickými operátory a dále řetězcové výrazy s operátory pro řetězce. Přehled základních operátorů:

- $a > b$, $a < b$, $a \geq b$, $a \leq b$, $a = b$
- IS NULL, IS NOT NULL
- LIKE 'K%', LIKE 'L____', LIKE '_b%k'
- AND, OR, NOT,
- (,)

Projekce

Projekci si lze představit jako zobrazení konkrétní položky (sloupce). Jednotlivé řádky tabulky mohou vystupovat jako n-tice hodnot.

Jako seznam sloupců se uvedou právě projektované sloupce, tj. sloupce, jejichž hodnoty se mají vybrat. Může zde být uveden výčet sloupců z více tabulek. Pokud by se některé názvy sloupců v různých tabulkách opakovaly, pak je potřeba název sloupce uvést ve tvaru jméno_tabulky.jméno_sloupce. Jména sloupců se oddělují čárkou.

FROM seznam tabulek

V této části jsou uvedeny tabulky, ze kterých se mají data získávat. Všechny sloupce, zadané za klíčovým slovem SELECT musí patřit do zde uvedených tabulek. Názvy tabulek se oddělují také čárkou [14].

Příklady:

Názvy nakladatelství, která jsou z Brna:

```
SELECT název  
FROM nakladatelství  
WHERE adr_psč >= 60000  
      AND adr_psč <= 64400
```

Název a e-mail všech nakladatelství, která mají svou webovou stránku:

```
SELECT název, email  
FROM nakladatelství  
WHERE web_stránka IS NOT NULL
```

9.8 Komunikace PHP s MySQL

V případě připojení k MySQL databázi pomocí PHP skriptů, stává se klientem databáze PHP. Pokud chce PHP s databází pracovat, musí se k ní připojit. Po připojení může komunikovat s databázovým serverem, posílat mu SQL příkazy, přebírat získaná data, apod. Po ukončení práce PHP s databází se provede odpojení od databáze. Pro připojení k MySQL slouží v PHP funkce `mysql_connect`.

Tato funkce se obvykle používá se třemi parametry:

```
mysql_connect ( adresa serveru, uživatelské_jméno, uživatelské_heslo ).
```

Parametr `adresa serveru` je adresa databázového serveru MySQL. Samotná funkce `mysql_connect` vrací k dispozici tzv. identifikátor spojení, který se používá k posílání příkazů do MySQL. V případě, že se spojení nezdaří, funkce vrátí hodnotu `false`. Díky tomu se může v podmínce testovat, zda spojení s databází proběhlo v pořádku [1].

II. PRAKTICKÁ ČÁST

10 ÚKOL 1

Pomocí formulářových prvků se vytvoří stránka pro registraci údajů. Poté, co se data odešlou, zobrazí se zadané údaje.

Prvním krokem je vytvoření formuláře

```
1 <form action="Ukol1b.php" method="post">
2
3 Jméno: <input type="text" name="jmeno">
4 Příjmení: <input type="text" name="prijmeni">
5 E-mail: <input type="text" name="email">
6 <input type="submit" value="Odešli údaje">
7
8 </form>
```

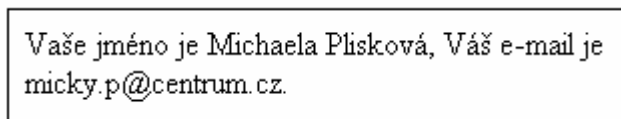


The screenshot shows a web form with three text input fields and a submit button. The first field is labeled 'Jméno:' and contains the text 'Michaela'. The second field is labeled 'Příjmení:' and contains the text 'Plisková'. The third field is labeled 'E-mail:' and contains the text 'micky.p@centrum.cz'. Below the fields is a button labeled 'Odešli údaje'.

Obr. 5 Prvky formuláře

V druhém kroku se vypíše zadané údaje

```
9 echo "Vaše jméno je $jmeno $prijmeni, Váš e-mail je $email.";
```



The screenshot shows a message box containing the text: 'Vaše jméno je Michaela Plisková, Váš e-mail je micky.p@centrum.cz.'

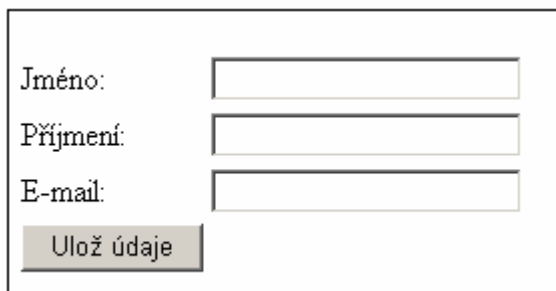
Obr. 6 Výpis zadaných údajů

11 ÚKOL 2

Pomocí 3 souborů se vytvoří jednoduchý adresář. V prvním souboru se zadávají údaje, které se mají uložit. Po odeslání údajů se načte druhý soubor. Ten otevře třetí soubor pro zápis a uloží do něj údaje. Obsahuje odkaz na otevření adresáře – třetího souboru.

První soubor

```
1 <form action="ulozit.php" method="post">
2
3 <table align="center">
4 Jméno: <input type="text" name="jmeno">
5 Příklad: <input type="text" name="prijmeni">
6 E-mail: <input type="text" name="email">
7 <input type="submit" value="Ulož údaje">
8
9 </table>
10 </form>
```



Obr. 7 Úvodní skript

Druhý soubor

```
1 $date=Date("j. " . "m. " . "Y, " . "H:i:s");
2 $fp=fopen("kontakt.php","a+");
3 $uloz="<TR>
4 <TD>$jmeno</TD>
5 <TD>$prijmeni</TD>
6 <TD>$email</TD>
7 <TD>$date</TD>
8 <TD><a href=mailto:$email><I>Napsat</I></a></TD>
9 </TR>\n";
```

```

10 fputs($fp,$uloz);
11 fclose($fp);
12 echo"Údaje byly úspěšně uloženy.<BR><BR>" ;
13 <A href="kontakt.php">Jdi na kontakty</A>

```

Údaje byly úspěšně uloženy.

[Jdi na kontakty](#)

Obr. 8 Druhý skript

Třetí soubor

```

1 echo"<body bgcolor=#DCDCDC ><table border=1 width=60%
2 bgcolor=white cellpadding=2
3 <TD><I><b>Jméno</b></I></TD>
4 <TD><I><b>Přijmení</b></I></TD>
5 <TD><I><b>E-mail</b></I></TD>
6 <TD><I><b>Datum</b></I></TD>
7 <TD><I><b>Kontakt</b></I></TD></TR>" ;?>

```

<i>Jméno</i>	<i>Přijmení</i>	<i>E-mail</i>	<i>Datum</i>	<i>Kontakt</i>
Tomáš	Holub	t_holub@centrum.cz	14. 11. 2004, 21:20:41	Napsat
Jan	Kocourek	kocicka@post.cz	14. 11. 2004, 21:23:34	Napsat
Markéta	Jurinová	jurina@centrum.cz	14. 11. 2004, 21:25:11	Napsat

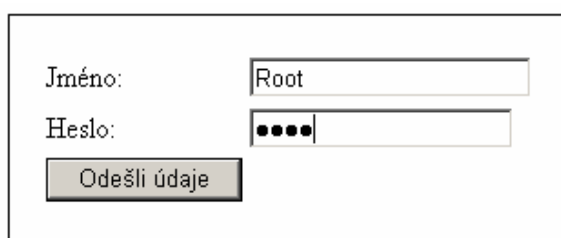
Obr. 9 Výsledný adresář

12 ÚKOL 3

Pomocí cookies se vytvoří jednoduché počítadlo návštěvnosti. Cookie je platná 60 sekund, poté se vynuluje a začne počítat znovu.

Přihlášení pod uživatelem Root a heslem Root

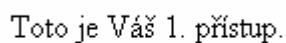
```
1 <form action="2.php" method="post">
2 Jméno: <input type="text" name="jmeno">
3 Heslo: <input type="password" name="heslo" size="20" >
4 <input type="submit" value="Odešli údaje">
5 </form>
```



Obr. 10 Prvky formuláře

Spuštění skriptu s cookies

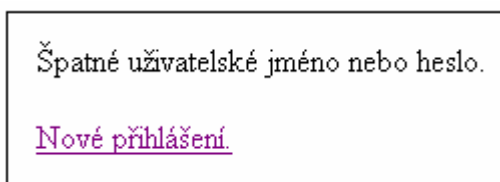
```
1 if ($jmeno == "Root" AND $heslo == "Root") {
2 SetCookie("pocitej_n", ++$pocitej_n, time()+60);
3     echo "Toto je Váš $pocitej_n. přístup."; }
```



Obr. 11 Počítadlo přístupů

Zadání špatného uživatelské jména nebo hesla

```
4 else { echo "Špatné uživatelské jméno nebo heslo.";
5 <A href="1.php">Nové přihlášení.</A> }
```



Obr. 12 Špatné údaje

13 ÚKOL 4

Pomocí řádkového klienta mysql se vytvoří databáze a v ní jedna tabulka.

Vytvoření databáze

```
1 CREATE DATABASE Zkouska;  
2 USE Zkouska;
```

Vytvoření tabulky

```
3 Create table Zaznam(  
4   zaznam integer primary key auto_increment,  
5   jmeno varchar(30),  
6   email varchar(30),  
7   telefon varchar(9),  
8   poznamka text);
```

Přidání jednoho řádku do tabulky

```
9 Insert into Zaznam VALUES(1,"Martin  
Novy",mano@email.cz,"728945368","Felix");
```

Zaznam	Jmeno	Email	Telefon	Poznamka
1	Martin Novy	mano@email.cz	728945368	Felix
2	Petr Kubik	pkub@email.cz	726458124	Fotbal

Obr. 13 Výsledná tabulka

14 ÚKOL 5

V předchozím úkolu byla vytvořena databáze. Nyní se tabulka z této databáze zobrazí v internetovém prohlížeči.

Nejdříve se vytvoří připojení k databázi

```
1 $spojeni = mysql_connect('localhost','root','root');
```

Následuje vybrání databáze

```
2 $vysledek = mysql_select_db('Zkouska',$spojeni);
```

Poté se musí nadefinovat proměnné

```
3 $Jmeno = $_POST['Jmeno'];  
4 $Email = $_POST['Email'];  
5 $Telefon = $_POST['Telefon'];  
6 $Poznamka = $_POST['Poznamka'];
```

Výběr všech záznamů z tabulky

```
7 $sql = "SELECT * FROM Zaznam";  
8 $vysledek = mysql_query($sql,$spojeni);
```

Zobrazení záznamů z tabulky

```
9 While ($radek = mysql_fetch_row($vysledek))  
10 { echo '<tr>'; for ($i=0; $i<5; ++$i)  
11     echo '<td>', $radek[$i], '</td>';  
12     echo '</tr>'; }
```

Ukončení spojení

```
13 mysql_close($spojeni);
```

Záznam	Jméno	Email	Telefon	Poznámka
1	Martin Novy	mano@email.cz	728945368	Felix
2	Petr Kubik	pkub@seznam.cz	726458124	Fotbal
3	Karel Musil	kaja@email.cz	721569324	Hokej

Obr. 14 Výsledná tabulka v prohlížeči

15 SYSTÉM S UDRŽOVÁNÍM KONTEXTU

Jednoduchý internetový obchod s knihami. Při spuštění se zobrazí úvodní strana.

Systém - jednoduchý internetový obchod s knihami

Zadejte uživatelské jméno a heslo.
Nový uživatel se musí nejdříve zaregistrovat.

Uživatelské jméno:

Heslo:

[Registrace nového uživatele](#)

Obr. 15 Úvodní strana

Neregistrovaní uživatelé se musí nejdříve zaregistrovat, vyplnit dané údaje.

Zaregistrujte se prosím:

Jméno:

Uživatelské jméno:

Heslo:

Potvrzení hesla:

Adresa:

Město:

Telefonní číslo:

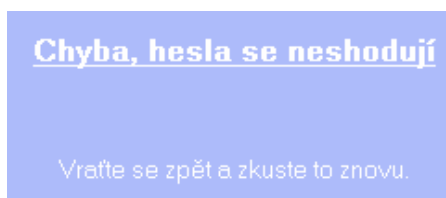
Obr. 16 Registrace

Pokud jsou zadány správné údaje, zobrazí se zpráva o úspěšné registraci.



Obr. 17 Správná registrace

Pokud se zadaná hesla neshodují, vypíše se chyba.



Obr. 18 Špatná hesla

Nyní se zadává uživatelské jméno a heslo pro vstup do systému. Pokud je přihlašovací jméno nebo heslo špatné, zobrazí se chybové hlášení.

A blue rectangular box containing a login form. At the top, it says "Špatné přihlašovací jméno nebo heslo." in all caps. Below it, in smaller text, it says "Nový uživatelé se musí nejdříve zaregistrovat." The form has two input fields: "Uživatelské jméno:" and "Heslo:". Below the "Heslo:" field is a "Vstup" button. To the right of the form, there is a purple link that says "Registrace nového uživatele".

Obr. 19 Špatné přihlášení

Po vstupu do systému se zobrazí nabídka knih.

Nabídka knih	
Harry Potter a Tajemná komnata J. K. Rowlingová pro děti \$350	Do košíku
Harry Potter a Kámen mudrců J. K. Rowlingová pro děti \$430	Do košíku
CSS Eric Meyers počítačová lit. \$290	Do košíku
Technické prostředky automatického řízení J. Balátě a kol. technická lit. \$400	Do košíku

Obr. 20 Nabídka knih

Pokud si uživatel vybere nějakou knihu, kliknutím na „Do košíku“, danou knihu přidá do nákupního košíku.

Přidáno do košíku	
CSS počítačová lit. \$290	Zobrazit košík

Obr. 21 Výběr knihy

Kliknutím na „Zobrazit košík“ se zobrazí knihy, které byly vybrány.

Košík					
Číslo	Název	Autor	Žánr	Množství	Cena
1	CSS	Eric Meyers	počítačová lit.	2	290 Kč

Změna množství
Objednej

Obr. 22 Košík

Po potvrzení se zobrazí objednávka s celkovou cenou a adresou, na kterou mají být knihy odeslány.

Číslo	Název	Autor	Žánr	Množství	Cena
1	CSS	Eric Meyers	počítačová lit.	2	290 Kč

Celková cena: 580 Kč

Zadané údaje

Jméno: Martin Nový
Adresa: Jedovnice 75
 Blansko

Obr. 23 Objednávka

Nabídka „Stav účtu“ zobrazí všechny objednané knihy a celkovou částku.

Objednávka				<u>Stav účtu: 580 Kč</u>	
Číslo	Název	Autor	Množství	Datum	
88	CSS	Eric Meyers	2	2005-05-21	

Obr. 24 Stav účtu

ZÁVĚR

Hlavním cílem této bakalářské práce bylo vytvořit studijní materiály pro předmět Technologie WWW.

Práce je rozdělena na praktickou a teoretickou část. V teoretické části je uvedena teorie PHP a databáze MySQL. V praktické části jsou uvedeny jednoduché příklady.

Při vypracování této bakalářské práce jsem se seznámila s programovacím jazykem PHP a naučila jsem se vytvářet jednoduché aplikace. Také jsem se seznámila s databází MySQL.

Pro vytváření studijních materiálů bylo nutné mít na počítači nainstalovaný nějaký webový server. Použila jsem Uniserver, který je volně šířitelný. Jeho součástí je webový server Apache i databáze MySQL. Vytvořené materiály byly umístěny na výukový systém MOODLE, běžící na serveru vyuka.ft.utb.cz.

Veškeré vypracované materiály a kompletní zdrojové kódy k praktické části jsou uvedeny na CD.

SEZNAM POUŽITÉ LITERATURY

- [1] PHP krok za krokem. Dostupné z: <<http://www.builder.cz>>.
- [2] Kurz PHP. Dostupné z: <<http://www.zive.cz>>.
- [3] Interpreter PHP. Dostupné z <<http://www.php.cz>>.
- [4] Webový server Apache. Dostupné z <<http://www.apache.org>>.
- [5] Vývoj aplikací. Dostupné z <<http://www.interval.cz>>.
- [6] Manuál PHP. Dostupné z <<http://www.php.cz>>.
- [7] CASTAGNETTO J., RAWAT H., SCHUMANN S., SCOLLO CH., VELIATH D.
Programujeme PHP profesionálně 2. vyd. Praha: Computer Press, 2002. ISBN 80-7226-310-2.
- [8] Studijní materiály předmětu Databázové systémy.
- [9] Seriál PHP. Dostupné z: <<http://www.linuxsoft.cz>>.
- [10] Manuál MySQL. Dostupné z: <<http://mm.gene.cz>>.
- [11] Databázový server. Dostupné z <<http://www.mysql.cz/>>.
- [12] PHP. Dostupné z <<http://www.tvorba-webu.cz>>.
- [13] PHP od začátku. Dostupné z <<http://www.webguru.cz>>.
- [14] Vývoj aplikací, databáze. Dostupné z <<http://www.interval.cz>>.
- [15] PHP, seriály. Dostupné z <<http://jaknaweb.com>>.
- [16] Manuál MySQL. Dostupné z <<http://www.slunecnice.cz>>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PHP	skriptovací jazyk
HTML	jazyk pro tvorbu WWW stránek
ODBC	čerpání dat např. z MS Access nebo z MS Excelu.
HTTP	protokol, definující pravidla komunikace mezi klientem a serverem
FTP	protokol pro přenos souborů
POP3	služba pro stahování zpráv
SMTP	protokol sloužící k odesílání pošty
URL	popis umístění souboru v síti
TCP/IP	komunikační protokol
DBMS	system řízení báze dat
SŘBD	system Řízení Báze Dat
SQL	jazyk pro programování databází
DDL	příkazy pro definici a modifikaci databázových objektů.
DML	příkazy pro manipulaci s databázovými údaji
WWW	služba Internetu

SEZNAM OBRÁZKŮ

Obr. 1 Relace 1:1	32
Obr. 2 Relace N:1	32
Obr. 3 Relace N:M.....	33
Obr. 4 Řádkový klient MySQL.....	41
Obr. 5 Prvky formuláře	48
Obr. 6 Výpis zadaných údajů.....	48
Obr. 7 Úvodní skript	49
Obr. 8 Druhý skript	50
Obr. 9 Výsledný adresář	50
Obr. 10 Prvky formuláře	51
Obr. 11 Počítadlo přístupů.....	51
Obr. 12 Špatné údaje.....	52
Obr. 13 Výsledná tabulka	53
Obr. 14 Výsledná tabulka v prohlížeči	54
Obr. 15 Úvodní strana.....	55
Obr. 16 Registrace	55
Obr. 17 Správná registrace	56
Obr. 18 Špatná hesla	56
Obr. 19 Špatné přihlášení	56
Obr. 20 Nabídka knih.....	57
Obr. 21 Výběr knihy	57
Obr. 22 Košík.....	57
Obr. 23 Objednávka.....	58
Obr. 24 Stav účtu	58

SEZNAM TABULEK

Tab. 1 Únikové sekvence.....	11
------------------------------	----

SEZNAM PŘÍLOH

P 1 : Přenosné médium CD s vytvořenými materiály