Přístup uživatelů do databáze MP Kyjov přes webové rozhraní

Access for users into database of MP Kyjov through web interface

Bc. Martin Vaculík

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

nascannované zadání s. 1

ABSTRAKT

Diplomová práce má uživateli, prostřednictvím webového rozhraní, nabídnout možnost přístupu do databází Městské policie Kyjov. Databáze uchovávají informace o nalezených věcech a nahlášených ztrátách. Uživatelům se informace zobrazují na základě definovaných dotazů, které jsou různé pro dané skupiny uživatelů. Zprostředkování komunikace mezi internetovým prohlížečem a databází je prováděno pomocí PHP skriptů.

Klíčová slova: databáze, webové rozhraní, ztráty, nálezy, Městská policie Kyjov

ABSTRACT

This diploma work should offer the user a posibility of access to a database of MP Kyjov using the web interface.Databases keep information about lost properties and losses. The information is displayed on the basis of defined inquires, which vary from the group of users.Mediation of information between an internet viewer and a database is carried out with a help of PHP scripts.

Keywords: database, web interface, losses, findings, Kyjov town police

Děkuji své vedoucí diplomové práce Ing. Zdence Prokopové CSc., za prvotřídně odvedenou práci, kterou mi věnovala již při psaní bakalářské práce a nyní i práce diplomové. Ještě jednou jí tímto mnohokrát děkuji.

Děkuji také své rodině, svým blízkým, kteří mi věřili a podporovali při studiu bakalářském a posléze i magisterském.

Zvláště chci poděkovat mojí manželce Hance za to, že po celou dobu mého studia při mne stála v dobrém i zlém a snažila se mi pomoci jak nejlépe uměla. Celé ty roky mi byla obrovskou oporou a jedině jí patří poděkování za to, že jsem došel až do konce.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

OBSAH

ÚVOD

Při výběru tématu diplomové práce jsem chtěl vytvořit něco, co bude přínosem a bude dále využíváno v praxi. Napadlo mě několik možných témat, ale nakonec jsem vybral aplikaci, která bude navazovat na moji bakalářskou práci. Bakalářská práce byla zaměřena na vytvoření nové webové prezentace Městské policie Kyjov s důrazem kladeným na poskytnutí možnosti vzájemné komunikace Městské policie Kyjov s občany. Tato práce měla ohromný úspěch a proto mi bylo jasné, že bych měl občanům nabídnout nějakou novou službu. Při jejím výběru mi pomohli samotní občané, kteří své požadavky prezentovali právě přes webové stránky. Požadovali v největší míře možnost prohlédnout si seznam nalezených věcí. S návrhem souhlasilo vedení městské policie spolu s představiteli města Kyjov. Celý záměr jsem představil paní Ing. Zdence Prokopové, CSc., která záměr schválila s tím, že se rozšíří o možnost vkládání nahlášených ztrát. Výběr záměru je popsán v teoretické části diplomové práce. Dále tato část obsahuje návrh budoucího systému uložení a následného prezentování informací uživatelům. Návrh představuje databázi uloženou na serveru, která bude obsahovat všechna potřebná data. Jednotlivá data budou uložena v tabulkách databáze, podle toho, jaké záznamy popisují a také podle toho, jak moc jsou důvěrná. Návrh se ještě zabývá otázkou dotazů na databázi, které budou kladeny jednotlivými skupinami uživatelů.

Praktická část diplomové práce již řeší samotný postup vytvoření kompletní databáze. Nejdříve je databáze vytvořena a k ní uživatelské účty s definovanými oprávněními. Následuje vytvoření všech potřebných tabulek pro uložení informací o nálezech a ztrátách. Praktická část diplomové práce v kapitole č.5 se zabývá otázkou: „Jak řešit propojení databáze s webovým rozhraním?“. Je zde popsán způsob komunikace databáze s webovým rozhraním prostřednictvím skriptů jazyka PHP. Nejprve je popsán způsob vytvoření připojení k databázi pro jednotlivé skupiny uživatelů. Dále jsou definovány dotazy na databázi MySQL implementované do PHP skriptů tak, aby je bylo možné vykonat z webového rozhraní. Je vytvořen formulář pro možnost nahlášení ztráty uživatelům prostřednictvím webové prezentace Městské policie Kyjov.

Závěr práce je věnován bezpečnosti systému. Bezpečnost je velmi důležitá, zvlášť pokud jde o data, která by mohla být zneužita. Bohužel stoprocentně bezpečný systém neexistuje, ale snažil jsem se vytvořit práci tak, abych útočníkovi práci v žádném případě neusnadnil.

TEORETICKÁ ČÁST

1 ANALÝZA POŽADAVKŮ NA DATABÁZOVÝ SYSTÉM

Při návštěvě webové prezentace Městské policie Kyjov vám je nabídnuta možnost, vyjádřit se k jejímu obsahu a navrhnout nové podněty pro její rozšíření. Prezentace je v provozu již téměř dva roky a za tuto dobu bylo přijato několik desítek návrhů a připomínek na její obměnu a rozšíření. Zhruba 70% podnětů bylo jen upozornění na různé nesrovnalosti v textu, nebo na lépe vystihující odkaz než byl původní s danou problematikou. Zbývajících 30% podnětů se týkalo nejrůznějšího rozšíření prezentace. Největší část tvořil požadavek na podrobnější popis výcviku služebních psů, práci s nimi a možnost objednání ukázky výcviku psů přímo u objednatele. Tento požadavek musel být velitelem zamítnut, neboť vytíženost služebních psů a jejich psovodů je již tak velmi vysoká. Další přání uživatelů se týkala zveřejnění našich pracovních zákroků, spíše z oblasti těch zajímavých, výstižněji řečeno úsměvných. Tento požadavek se snažíme průběžně realizovat, s přihlédnutím k omezení, které nám ukládá zákon. Podstatně zajímavější podněty byly z oblasti usnadnění nalezení věcí, které lidé někde ztratily. Tyto podněty nebyly zastoupeny v tak velké míře jako předchozí. Je to dáno především tím, že ne každý se již s touto nepříjemností setkal. V případě že ano, zřejmě jej ani nenapadlo informovat se také na městské (obecní) policii v daném místě.

1.1 Podněty strážníků městské policie

Městská policie Kyjov, tak jako zřejmě všechny ostatní obecní (městské) policie, skladuje nalezené věci v prostorech k tomu určených. Tyto věci zajišťují přímo strážníci při plnění svých služebních povinností, ale ve větší míře jsou věci přinášeny přímo občany města, kteří je přinesou odevzdat jako nález. Nelezené věci se evidují v knize nálezů, kde se spolu s přidělením evidenčního čísla přikládá podrobný popis nalezené věci. Věc je dále nafocena a uložena do skladu. U věcí kde je znám majitel (osobní doklady, registrované jízdní kolo, apod.) je majitel ihned vyrozuměn o jejím nalezení a vyzván k jejímu odběru. V případě věcí kde tomu tak není, jsou věci po určitou dobu skladovány a po vypršení této doby jdou do veřejné dražby jak je stanoveno zákonem. V největším počtu ze všech nalezených věcí jsou zastoupeny jízdní kola. Jejich majitelé ve většině případů ani netuší, že je jejich kolo již nalezeno a oni si jej mohou jednoduše vyzvednout. Vydání majiteli je uskutečněno na základě předložení dokladu o nabytí jízdního kola, nebo při uvedení jeho podrobného popisu. Tento postup vydání se vztahuje na všechny nalezené věci. Všichni

strážníci Městské policie Kyjov měli prostor pro vyjádření svých návrhů a připomínek na interních poradách městské policie, opakujících se pravidelně několikrát v roce. Interní porady probíhají za účasti představitelů města Kyjova. Ti na základě informací z besed s občany pojmenovaných: „Hovory s veřejností“, předložili své vlastní návrhy na rozšíření webové prezentace Městské policie Kyjov.

1.2 Zhodnocení požadavků a výběr návrhu

Všechny získané podněty, od samotných občanů, představitelů města Kyjov, strážníků Městské policie Kyjov a jejího velitele, byly několikrát důkladně rozebrány a projednány se všemi stranami. Nejvíce hlasů se přiklánělo k volbě vytvoření databáze nalezených věcí a jejímu zveřejnění prostřednictvím webového rozhraní na stránkách Městské policie Kyjov. Pro tuto volbu nahrávala také okolnost, že všechny úložné prostory s nalezenými věcmi jsou již dost zaplněné. Proč tedy nenabídnout lidem možnost, jak získat své věci zpět. Tento návrh jsem předložil své vedoucí diplomové práce Ing. Zdence Prokopové, CSc., jako možnost na rozšíření webové prezentace Městské policie Kyjov. Ta s volbou návrhu rozšíření webové prezentace souhlasila a zároveň navrhla doplnění databáze ještě o možnost vkládání nahlášených ztrát od jednotlivých uživatelů. Toto doplnění je velmi vhodné, jelikož je úzce spjato s vybraným návrhem.

2 NÁVRH STRUKTURY DATABÁZE

Nejdůležitějším krokem v návrhu celé databáze, byla správná volba jednotlivých tabulek. Tabulka je základním stavebním kamenem pro budování celé databáze. Skládá se ze sloupců a řádků. Tabulka nám popisuje nějakou entitu. Jednotlivé řádky tabulky, říkáme jim záznamy, reprezentují prvky dané entity. Sloupce tabulky, neboli atributy (pole), volíme podle toho jaké vlastnosti nás o dané entitě zajímají. Každý řádek tabulky obsahuje množinu hodnot, které náleží konkrétním sloupcům. Každá hodnota musí mít datový typ specifikovaný pro daný sloupec.

Dalším důležitým bodem návrhu databáze je předem dobře promyslet jaké dotazy budeme asi na databázi klást. Jaké dotazy budou použity pro přehledné a efektivní vyhledávání informací z databáze. Jistě budou použity jiné dotazy pro běžné uživatele a uživatele s přístupovými právy. Těm bude zpřístupněn mnohem větší okruh informací z databáze a také poskytnuta možnost vkládání nových údajů do databáze. Pro všechny tyto operace budou vytvořeny patřičné dotazy, které by měli efektivně plnit zadané úkoly. Proto je nutné navrhnout databázi tak, aby se při každém dotazu nemuselo prohledávat zbytečně velké množství informací uložených v databázi.

Také prázdná pole v databázi nejsou vhodná. Není dobré pokud má pole v mnoha řádcích tabulky prázdná místa, tedy žádnou (známou) hodnotu (hodnotu null). Pokud máme v databázi hodně hodnot null, tak je to jednak plýtvání místem a navíc to způsobuje potíže například při provádění výpočtů ve sloupci. Pokud uživatel uvidí v tabulce hodnotu null, nepozná, zda to znamená, že je daný atribut irelevantní, nebo zda je v databázi chyba, nebo zda prostě ještě hodnota nebyla pořízena. Použitím vyplývajících z velkého počtu hodnot null se vyhneme, když zvolíme alternativní návrh databáze. Ve většině případů jde o způsob vytvoření další nové tabulky s patřičnými klíči a relacemi. [4]

Moje práce bude obsahovat databázi, která vyšla z návrhů všech stran jako nejžádanější. Bude se jednat o databázi pojmenovanou jako: „ztraty_nalezy“. Ještě než budu moci databázi vytvořit, musím si nejdříve nainstalovat patřičný systém a přihlásit se do něj jako uživatel „root“. Přidělím uživateli root heslo, protože při instalaci MySQL přihlašovací heslo není vytvořeno. Dále vytvořím uživatelský účet „administrator“, kterému přidělím všechna práva. Je to z důvodu, aby uživatel root nebyl využíván pro běžné operace, ale jen pro správu databáze. Nakonec vytvořím uživatelské účty pro samotné uživatele webových

stránek Městské policie Kyjov. Těmto uživatelům přidělím potřebná oprávnění, která jim umožní prohlížení informací z databáze Městské policie Kyjov přes webové rozhraní. Uživatelské účty budou tři a budou se lišit právě přidělenými právy. To uživatelům umožní provádět v databázi rozdílné operace a prohlížení více či méně důležitých informací. První uživatelský účet pojmenuji: „vsichni“. Tento uživatel bude principu nejnižšího uživatelského práva. Tedy nebude mít možnost provádět žádné operace v databázi, jen mu bude povoleno prohlížení informací a to ještě ze striktně daných tabulek. Druhý uživatel již bude mít práva rozšířenější, ale opět nebude moci provádět operace na databázi či jednotlivých tabulkách. Bude mít oproti prvnímu uživateli možnost prohlížet data ze všech tabulek databáze. Tento účet bude pojmenován: „straznik“ a bude mu přiděleno přihlašovací heslo. Posledním třetím uživatelským účtem bude: „spravce“. Ten již bude mít možnost provádět různé operace na databázi a jejích tabulkách. Prohlížení všech uložených informací v databázi je samozřejmé. Účet bude zabezpečen přihlašovacím heslem.

Předem jsem kontaktoval technika serveru, kde je umístěna webová prezentace Městské policie Kyjov a domluvili jsme se na požadovaných službách. V mém případě se jednalo služby MySQL a PHP. Obě dvě jsou serverem poskytovány a není potřeba dodatečná úprava smlouvy mezi městem a poskytovatelem. Dohodli jsme se také s technikem na vytvoření již zmíněných uživatelských účtů a databázi, protože tato možnost je umožněna jen administrátorovi daného serveru. Při přípravě mojí práce pracuji na svém stroji, proto jsem administrátorem sám. Jakmile budu mít práci připravenu, bude přenesena na server.

2.1 Návrh tabulek

Ve spojitosti s dobře navrženými tabulkami se používá pojem: „Normální formy“. Správně navržené tabulky splňují čtyři základní normální formy:

První nejjednodušší normální forma (1NF) říká, že všechny atributy tabulky jsou atomické. Znamená to, že jsou dále již nedělitelné.

Druhou normální formu (2NF) tabulka splňuje, právě když splňuje 1NF a navíc každý atribut, který není primárním klíčem, je na primárním klíči plně závislý. To znamená, že se v řádku tabulky nesmí objevit položka, která by byla závislá jen na části primárního klíče. Z definice vyplývá, že problém 2NF se týká jenom tabulek, kde volíme za primární klíč více položek než jednu.

Tabulka splňuje třetí normální formu (3NF), jestliže splňuje 2NF a žádný atribut, který není primárním klíčem, není tranzitivně závislý na žádném klíči. Tranzitivně závislý znamená, že atribut1 je závislý na atributu2 a tento atribut2 je závislý na primárním klíči. Potom díky tranzitivitě dostaneme závislost atributu1 na primárním klíči.

Poslední formou je tzv. Boyce-Coddova normální forma (BCNF). Tabulka splňuje BCNF, právě když pro dvě množiny atributů A a B platí, že B je závislé na A a zároveň B není podmnožinou A, pak množina A obsahuje primární klíč tabulky. Tato forma zjednodušuje práci s tabulkami a pokud při tvorbě tabulek postupně splňujeme 1NF, 2NF a 3NF, splníme i formu BCNF. [5]

Dobře navrhnout tabulky je klíčový a důležitý úkol zejména z dlouhodobého hlediska. Pokud bychom v databázovém systému, který již bude běžet v ostrém provozu nalezli chybu spočívající ve špatně navržené databázi, mělo by to katastrofální důsledky.

Právě proto jsem návrhu jednotlivých tabulek věnoval nejvíce času a úsilí. Postupně vzniklo několik verzí návrhů tabulek, které jsem několikrát přehodnocoval a konzultoval s mojí vedoucí diplomové práce Ing. Zdenkou Prokopovou, CSc..

Nejdříve jsem si stanovil jaká data budou v databázi uložena. Zdrojem informací mi byly vlastní pracovní zkušenosti, ale především přehled předchozích záznamů o ztracených a nalezených věcech, nebo zvířatech. Všechny tyto data potřebné k identifikaci jednotlivých záznamů je nutné začlenit do databáze. Jen tak může databáze sloužit svému účelu.

Nyní, když mám nadefinována data potřebná pro databázi, musím je rozčlenit do jednotlivých tabulek databáze. Tyto data jsou dvojího typu. Jedny jsou určeny všem, řekněme že jsou veřejná a druhá jsou určena jen pro oprávněné osoby a ty budou soukromá. Rozčlenění provedu takovým způsobem, aby data veřejná byla obsažena v jedné tabulce a data soukromá byla obsažena v druhé tabulce.

Jelikož je u některých nálezů či ztrát popisných informací více než u jiných např. jízdní kolo, je nutné vytvořit tabulek více, ale opět s rozdělením na tabulky obsahující data veřejná a nebo data soukromá. Tyto tabulky budou obsahovat detaily k jednotlivým záznamům a budou tak doplňovat tabulky s všeobecnými informacemi, které jsou u jiných záznamů dostačující. Tím se vyhneme vytváření prázdných polí v databázi. Při návrhu tabulek jsem také dbal na dodržení všech normálních forem pro správné vytvoření tabulek.

Nakonec vznikla za přispění mojí vedoucí diplomové práce Ing. Zdenky Prokopové, CSc., konečná verze návrhu tabulek databáze. Databáze: „ztraty_nalezy“ bude obsahovat sedm tabulek s daty potřebnými pro identifikaci jednotlivých záznamů. Čtyři tabulky budou určeny pro záznamy s nálezy a tři pro záznamy se ztrátami. Důvod je velmi prostý. Ztráty i nálezy obsahují stejný počet popisných dat, vlastně téměř identický. Rozdíl je ale v tom, že informace o nálezech nemohou být zveřejněny v takové míře, jako informace o ztrátách. Informace o nálezech musí být chráněny, aby nedošlo k jejich zneužití a vydání dané věci neoprávněné osobě. Proto jsou jednotlivá data o nálezech rozdělena do čtyř tabulek a jen dvě tabulky s daty budou určeny široké veřejnosti. Naproti tomu data o ztrátách budou široké veřejnosti zveřejněna ve velké míře a proto jen jedna tabulka ze tří určených pro záznamy ztrát bude obsahovat soukromá data.

Tabulky pro záznamy popisující jednotlivé nálezy jsou následující:

- nalezy_verejne

  Tabulka obsahuje sloupec s evidenčním číslem, které je jedinečné pro každý jednotlivý záznam. Dále sloupec s popisem dané věci, datumem nalezení věci a sloupcem přiřazujícím věcem značku podle jejich druhu. Tento sloupec se značkou je navíc indexován pro rychlejší vyhledávání věcí požadovaného druhu. Tabulka obsahuje všeobecná data a je určena široké veřejnosti.

- nalezy_verejne_detail

  Tato tabulka obsahuje podrobnější informace o nalezených věcech a doplňuje tak předchozí tabulku: „nalezy_verejne“. Tabulka opět obsahuje sloupec s evidenčním číslem a dále následují sloupce s číslem rámu a druhem jízdního kola. Jsou zde ještě obsaženy sloupce s označením barvy nalezených věcí a fotografií všech věcí mimo jízdních kol. To je z důvodu, že fotografie jízdních kol nebudou přístupny široké veřejnosti.

- nalezy_soukrome

  Již z názvu tabulky vyplívá, že informace z této tabulky jsou určeny jen pro oprávněné osoby. Tabulka je určena pro všeobecné informace o nalezených jízdních kolech. V tabulce je opět zastoupen sloupec s evidenčním číslem, typem, výrobcem, značkou a fotografií jízdního kola.

- nalezy_soukrome_detail

Tabulka obsahuje podrobnější informace, které jsou soukromé. Pracovat s nimi mohou jen oprávněné osoby. Tabulka obsahuje sloupec s evidenčním číslem, místem nálezu věci, jménem, příjmením, adresou a kontaktem na nálezce věci. Informace o nálezci nejsou povinné, ale dle zákona přísluší nálezci 10% z ceny nalezené věci. Tu by nálezci měl vyplatit majitel věci, pokud se o ni přihlásí. Někteří nálezci peníze nechtějí, ale chtějí být informováni o tom, zda věc byla předána jeho majiteli. V tabulce jsou ještě dva sloupce. Jeden s názvem specifický znak popisuje věc, která má nějaký jedinečný, nebo neobvyklý znak. Například ornament, samolepku, nápis atd.. Druhým a zároveň posledním sloupcem je stav. Tento sloupec uchovává informaci o aktuálním stavu nalezené věci. Zda se věc nachází ve skladu nalezených věcí, nebo zda již byla předána majiteli, nebo zda ji má již zamluvenou a nemůže se pro ni dostavit.

Nyní si popíši tabulky, které obsahují záznamy o nahlášených ztrátách. Jde o tabulky:

- ztraty_verejne

Tabulka všeobecně popisuje nahlášené ztráty věcí nebo zvířat. Je prakticky totožná s tabulkou: „nalezy_verejne“, jen je navíc doplněna dalšími třemi sloupci. Obsahuje tedy sloupec s evidenčním číslem, které je jedinečné pro každý jednotlivý záznam. Dále sloupec s popisem dané ztráty, datumem ztráty věci a sloupcem přiřazujícím věcem značku podle jejich druhu. Tento sloupec je opět indexován pro rychlejší vyhledávání. Tři přidané sloupce rozšiřují tabulku o možnost vložení informace o barvě, specifickém znaku a fotografii ztracené věci nebo zvířete. Informace v tabulce jsou určeny široké veřejnosti.

- ztraty_verejne_detail

Tato tabulka doplňuje předchozí tabulku: „ztraty_verejne“ o podrobnější údaje o ztrátách a zároveň její data jsou určena k prohlížení všem. Tabulka obsahuje informace o ztracených jízdních kolech, ale v určitých případech může být použita i pro jiné ztráty. Obsahuje sloupce s evidenčním číslem, druhem, typem, číslem rámu, výrobcem a značkou jízdního kola.

- ztraty_soukrome

Informace z této tabulky jsou již určeny jen oprávněným osobám. Popisuje majitele ztracené věci nebo zvířete. Obsahuje sloupce s evidenčním číslem, jménem, příjmením, bydlištěm a kontaktem na majitele. Poslední sloupec tabulky popisuje stav ztráty. Tedy její uložení, vydání, nebo že nalezena nebyla.

2.2 Návrh dotazů na databázi

Abychom mohli s databází pracovat, musíme si definovat dotazy. Dotazy se budou lišit podle toho kdo k databázi přistupuje. Nejmenší oprávnění budou mít běžní uživatelé, větší oprávnění budou mít uživatelé s přístupovými právy. Ti si budou moci prohlédnout prakticky všechny informace uložené v databázi. A právě k tomu je potřebné definovat dotazy, které uživatelům uložené informace z databáze zobrazí.

Pro první uživatelský účet, označený jako: „vsichni“, bude definován dotaz pro zobrazení záznamů v konkrétním časovém období. Jednotkou pro zobrazení bude jeden kalendářní rok. Další definovaný dotaz bude možnost zobrazení jen určitého druhu věci, např. jízdní kolo, klíče, ostatní. V případě, že uživatel zvolí položku jízdní kolo, bude mu ještě nabídnuta možnost hledání jízdního kola podle jím zadaného čísla rámu, druhu a barvy jízdního kola. Implicitně se tomuto uživateli budou zobrazovat data jen z tabulek označených jako veřejné.

Všechny uvedené dotazy se vztahují k nalezeným věcem. V případě, že si uživatel bude chtít prohlížet nahlášené ztráty, bude mít navíc pro položku jízdní kolo větší možnost hledání dle parametrů. Bude mu nabídnuta možnost prohledat databázi podle zvoleného druhu, typu, barvy, výrobce a značky jízdního kola. Všechny ostatní dotazy vztahující se k nalezeným věcem má samozřejmě k dispozici také.

Uživatel s přiděleným přihlašovacím heslem pod označením: „straznik“, bude mít možnost využívat všechny dotazy jako uživatel: „vsichni“. Navíc bude mít možnost hledat nalezená jízdní kola podle výrobce, značky a typu. Může si zobrazit informace o místě nálezu a samotném nálezci. Pro ztráty je vše identické, jen místo nálezce si zobrazí informace o majiteli ztracené věci nebo zvířete. Implicitně se tomuto uživateli budou zobrazovat informace ze všech tabulek databáze.

Třetí uživatel označený jako: „spravce“, bude mít možnost využívat všechny definované dotazy a navíc možnost provádět různé operace na databázi a jejích tabulkách.

PRAKTICKÁ ČÁST

3 VYTVOŘENÍ DATABÁZE

Ještě než se pustíme do samotného vytvoření databáze, bylo nutné provést nezbytné operace, bez kterých by vytvoření databáze nebylo možné. V první řadě jde o instalaci MySQL. V mém případě pracuji na svém vlastním stroji a protože jsem provedl čistou instalaci je jediný uživatel, se kterým musím začít: „ROOT“. Pokud bych nebyl na svém vlastním stroji a nebo nebyl jeho administrátorem, musel bych požádat o vytvoření uživatelského účtu, pod kterým bych se do MySQL přihlásil. Administrátor také za nás musí vytvořit databázi a v ní nadefinovat již zmíněné uživatelské účty a jejich práva. Tato situace nastane při přenosu databáze na webový server, s jehož administrátorem jsem již domluven. Také je velmi vhodné používat stejnou verzi MySQL na vlastním stroji při vytváření databáze, jako je verze, kterou používá webový server, kde bude databáze umístěna. Verzi MySQL a zároveň také PHP můžeme zjistit kontaktováním správce webu, nebo přímo PHP dotazem na webový server. Jak jsem již uvedl pracuji na vlastním stroji a jsem přihlášen jako root. Po instalaci MySQL se vytvoří tabulky privilegií a do nich je zaznamenán uživatel root, který má neomezená privilegia. Jeho heslo je ale prázdné a tak se může kdokoli připojit jako root. Tento nebezpečný stav změním příkazem:

mysql> UPDATE user SET Password=PASSWORD('heslo_roota') WHERE user='root';
mysql> FLUSH PRIVILEGES;

V prvním řádku jsem přidělil heslo uživateli root. Protože se jednalo o přímý zápis do tabulky privilegií, musím tyto tabulky znovu načíst. To jsem provedl příkazem na druhém řádku. Dále si vytvořím uživatele s neomezenými právy jako má root. Jde o to, že root by měl být používán jen pro správu celé databáze. Vytvoření se provede:

*mysql> GRANT ALL PRIVILEGES ON *.* TO administrator@localhost IDENTIFIED BY 'administratorovo_heslo' WITH GRANT OPTION;*

Uživatel se jmenuje „administrator“ s heslem „administratorovo_heslo“. Administrátor má dovoleno se připojit jen z počítače, na kterém běží MySQL server. To je označeno "@localhost". Přidělil jsem mu veškerá práva "ALL PRIVILEGES" na všechny databáze a všechny tabulky v nich obsažené "*.*". Volba WITH GRANT OPTION dává administrátorovi možnost udělovat svá vlastní práva jiným uživatelům. Nyní se již přihlásím do sytému jako nově vytvořený uživatel administrator. Tyto kroky se shodují s kroky administrátora serveru kde bude databáze později umístěna. Samozřejmě by vše šlo udělat přímo

hned na webovém serveru, ale rozhodl jsem se pro tuto variantu. Nejprve musím tedy vytvořit databázi a nadefinovat jednotlivé uživatele, pro které bude databáze určena.

3.1 Vytvoření databáze, uživatelů a jejich práva

Databázový systém MySQL umí podporovat mnoho rozdílných databází. Obvykle se používá v jedné aplikaci jedna databáze. V mém případě se bude jednat také o jednu databázi s názvem: „ztraty_nalezy“. Pro vytvoření databáze použiji příkaz:

mysql> CREATE DATABASE ztraty_nalezy DEFAULT CHARACTER SET latin2 COLLATE latin2_czech_cs;

mysql>USE ztraty_nalezy;

Příkaz vytvoří databázi a zároveň definuje její znakovou sadu *latin2* se způsobem řazení dat *latin2_czech_cs*. To je důležitá věc, neboť se zobrazením češtiny jsou poměrně často problémy. Toto kódování jsem vybral záměrně. Důvodem je skutečnost, že webová prezentace Městské policie Kyjov používá právě toto kódování textu.

Druhým příkazem *USE* říkám, že chci pracovat s databází „ztraty_nalezy“.

Systém MySQL může mít mnoho uživatelů. Pro každého uživatele, který bude systém používat by měl být vytvořen účet a heslo. Heslo není povinné vytvářet všem uživatelům, ale doporučuje se nastavit ho. Pro účely webových aplikací bývá dobrým zvykem vytvořit alespoň jednoho uživatele na každou aplikaci. V databázi „ztraty_nalezy“ budou vytvořeny tři uživatelské účty.

Jedním z nejlepších rysů MySQL je, že obsahuje velmi sofistikovaný systém uživatelských práv. Uživatelské právo je oprávnění provádět konkrétní činnost s konkrétním objemem a je vázáno na konkrétního uživatele.

První uživatel bude principu nejnižšího uživatelského práva. Bude použit pro zvýšení bezpečnosti systému. Tento uživatel bude mít nejnižší úroveň práv, která mu ovšem ještě umožňuje provést patřičný úkol. Bude se jednat jen o možnost prohlížení informací a to ještě ne všech. Vytvořím jej příkazem:

mysql>GRANT SELECT ON ztraty_nalezy.nalezy_verejne, nalezy_verejne_detail, ztraty_verejne, ztraty_verejne_detail TO vsichni@"%";

Vytvořil jsem uživatele „vsichni“, který má privilegium jen k prohlížení informací z databáze „ztraty_nalezy“ a to jen z tabulky „nalezy_verejne, nalezy_verejne_detail, ztraty_verejne a ztraty_verejne_detail“, které později vytvořím. Uživatel se může přihlásit

z kteréhokoli počítače (@"%") bez zadávání přihlašovacího hesla. To jsem záměrně nenadefinoval, protože využívat tohoto uživatelského účtu budou běžní návštěvníci webové prezentace Městské policie Kyjov a vyplňování hesla není potřebné. Navíc toto heslo by muselo být na stránkách uvedeno. Tím by se stalo veřejným a ztratilo by svůj smysl.

Druhý uživatel bude mít práva o něco rozšířenější. Opět použiji příkaz GRANT:

mysql>GRANT SELECT ON ztraty_nalezy. TO straznik@"%" IDENTIFIED BY ' straznikovo_heslo';*

Tento uživatel s názvem „straznik“ již musí znát přihlašovací heslo, které jsem nadefinoval při vytvoření tohoto uživatele. Tím je mu umožněno prohlédnout si všechny dostupné informace ze všech tabulek obsažených v databázi „ztraty_nalezy“. Heslo pro přístup k tomuto uživatelskému účtu budou mít jen zaměstnanci Městské policie Kyjov. V současné době probíhají také intenzivní jednání se zástupci okolních obecních, městských policií našeho regionu, které byly osloveny a projevily zájem o tento projekt. Jejich zaměstnanci by tak získali přístup k informacím, které budou uloženy v databázi.
Třetí uživatel již bude mít práva velmi obsáhlá. Vytvořím uživatele: „spravce“ :

mysql>GRANT SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, CREATE, DROP, FILE ON ztraty_nalezy. TO spravce@"%" IDENTIFIED BY 'spravce_heslo';*

Tento třetí uživatelský účet poskytuje možnost prohlížení, vkládání, upravování, odstraňování, vytváření a načítání dat ze souborů. Opět je opatřen přihlašovacím heslem a možností přihlásit se z kteréhokoli počítače. Tento účet bude určen pro vybraného zaměstnance Městské policie Kyjov, který bude mít databázi na starosti. Tuto volbu jsem zvolil z důvodu, že ne každý zaměstnanec je natolik databázově gramotný, aby nemohl způsobit její poškození.

3.2 Tabulky databáze

Jak jsem již napsal v jedné z předchozích kapitol, nejdůležitějším krokem v návrhu celé databáze je správná volba tabulek. Tabulka je základním stavebním kamenem pro budování celé databáze. Pokud tedy již mám tabulky navrženy, mohu je vytvořit v mojí databázi. Databáze „ztraty_nalezy“ bude obsahovat sedm tabulek, které jsem navrhl v předchozí kapitole. Abych si udělal představu jak budou tabulky vypadat udělám si jednoduchý model tabulek v přehledném grafickém provedení. Nejprve se zaměřím na tabulky popisující nelezené věci.

První pod názvem: „nalezy_verejne“

Tabulka 1: nalezy_verejne

evc	popis	Datum	vec
150-07	jízdní kolo	2007.04.02	k
151-07	peněženka	2007.04.07	p
152-07	svazek klíčů	2007.04.17	kl
153-07	okapní plech	2007.04.25	os

Její vytvoření provedu příkazem jehož syntaxe je následující:

CREATE TABLE nalezy_verejne (evc VARCHAR (10) NOT NULL PRIMARY KEY, popis VARCHAR (60), datum DATE, vec ENUM ('k', 'p', 'kl', 'os'), INDEX (vec));

V příkaze pro vytvoření tabulky „nalezy_verejne“ vidíme jak jsou jednotlivé údaje zadávány. Nejprve je tabulka pojmenována a v závorce pak jsou nadefinovány jednotlivé její sloupce. Dále vidíme, že každý sloupec má za svým názvem uveden svůj datový typ. Ten je samozřejmě různý podle toho, jaká data bude sloupec obsahovat. Některé sloupce obsahují také další upřesňující parametry. Jde o parametr NOT NULL, který se používá tam, kde potřebujeme, aby v každé buňce byla zadána nějaká hodnota. PRIMARY KEY-takto označený sloupec bude sloužit jako primární klíč, tj. bude nám jedinečným způsobem identifikovat záznamy v tabulce. Datový typ ENUM nám dává možnost předem nadefinovat pole prvků, které mohou být vkládány do sloupce. Datový typ VARCHAR je obyčejný řetězec znaků a datový typ DATE nám umožňuje zadávat datum nálezu v pořadí "rok-měsíc-den". Naposled bych se zmínil o termínu INDEX. Ten je vázán na zadaný sloupec a tím můžeme jakoby „roztřídit“ jednotlivé položky uložené v tabulce do zvolených oddílů. Tabulka „nalezy_verejne“ má tyto oddíly čtyři. Jedná se o oddíl: „jízdní kolo“ označeno jako (k), dále „peněženka (p)“, „klíče (kl)“, „ostatní (os)“. Tyto oddíly nebyly zvoleny náhodou. Vyplynulo to z víceletého přehledu o tom, jaké jsou nejčastější nálezy evidovány na Městské policii Kyjov. Tímto indexováním dochází k jednoduššímu hledání v databázi. Například pokud bude chtít uživatel zobrazit jen jízdní kola, nemusejí se prohledávat všechny záznamy v databázi, ale stačí zobrazit jen položky indexovány pod označením (k).

Druhou tabulkou databáze je: „nalezy_verejne_detail“

Tabulka 2: nalezy_verejne_detail

evc	druh	barva	cislo_ramu	foto_ostatni
150-07	pánské	modré	xxxyyxxx	-
151-07	dámské	hnědá	-	151-07.jpg
152-07	-	zelené,modré	-	152-07.jpg
153-07	-	měděná	-	153-07.jpg

Syntaxe příkazu pro její vytvoření je následující:

CREATE TABLE nalezy_verejne_detail (evc VARCHAR (10) NOT NULL PRIMARY KEY, druh ENUM ('dámské', 'pánské', 'dětské'), barva VARCHAR (60), cislo_ramu VARCHAR (20), foto_ostatni VARCHAR (15));

V příkaze pro vytvoření tabulky vidíme stejné datové typy jaké byly použity u vytvoření předchozí tabulky. Ve sloupci: „foto_ostatni“ budou uloženy názvy fotografií všech nalezených věcí mimo jízdních kol. Samotné fotografie budou uloženy v adresáři na serveru a bude na ně vytvořen odkaz právě pomocí jejich názvů z tabulky. Můžeme si také všimnout, že u sloupce: „cislo_ramu“ je použit datový typ VARCHAR, což je řetězec znaků. Důvodem pro tento krok byla skutečnost, že se v číslech rámů jízdních kol vyskytují i písmena.

Třetí tabulkou k popisu nálezů je: „nalezy_soukrome“

Tabulka 3: nalezy_soukrome

evc	typ	vyrob- ce_znacka	foto
150-07	trek	favorit	150-07.jpg
154-07	silniční	velamos	154-07.jpg

Opět uvedu syntaxi pro vytvoření tabulky v MySQL:

CREATE TABLE nalezy_soukrome (evc VARCHAR (10) NOT NULL PRIMARY KEY, typ VARCHAR (20), vyrobce_znacka VARCHAR (40), foto VARCHAR (15));

V příkaze pro vytvoření tabulky se nám neobjevil žádný nový datový typ. Tabulka je určena k popisu nalezených jízdních kol. Její obsah si smí prohlížet jen oprávněný uživatel, který zná přihlašovací heslo. V tabulce jsou umístěny názvy fotografií nalezených jízdních kol. Je nutné je zabezpečit proti neoprávněnému použití cizích osob, protože vydání jízd-

ního kola je v některých případech podmíněno jen jeho detailním popisem. A to by měla osoba, po shlédnutí fotografie jízdního kola, velmi zjednodušené.

Poslední čtvrtou tabulkou popisující nálezy je: „nalezy_soukrome_detail“

Tabulka 4: nalezy_soukrome_detail

evc	misto_nalezu	nalez_jm	nalez_prij	nalez_kon	spec_znak	stav
150-07	Kollárova	Petr	Pan	777888999	nálepka	sklad
151-07	TGM	Tomáš	Fuk	518555333	ornament	vydáno

CREATE TABLE nalezy_soukrome_detail (evc VARCHAR (10) NOT NULL PRIMARY KEY, místo_nelezu VARCHAR (30), nalez_jm VARCHAR (15), nalez_prij VARCHAR (15), nalez_kon VARCHAR (20), spec_znak VARCHAR (40), stav ENUM ('sklad', 'vydáno', 'zamluveno', 'vyřazeno'));

Příkaz obsahuje nám již známé datové typy. Tabulka je zaměřena na osobu, která je nálezcem věci a souhlasila s uvedením osobních údajů a kontaktem pro možné pozdější vyrovnání s majitelem věci. V tabulce je také zaevidováno místo nálezu věci a současný stav nalezené věci. Tedy zda je věc uložena ve skladu, nebo již byla vydána majiteli. Může také dojít k jejímu vyřazení dle příslušných zákonů.

Nyní přistoupím k dalším tabulkám, které jsou v databázi: „ztraty_nalezy“ obsažené. Jde o tabulky, které popisují nahlášené ztráty věcí nebo zvířat. Tabulky jsou celkem tři.

První z nich je tabulka: „ztraty_verejne“

Tabulka 5: ztraty_verejne

evc	popis	datum	vec	barva	spec_znak	foto
200-07	jízdní kolo	2007.05.02	k	modrá	nálepka	200-07.jpg
201-07	peněženka	2007.05.07	p	hnědá	ornament	201-07.jpg
202-07	svazek klíčů	2007.05.17	kl	zelená	přívěšek	-
203-07	telefon	2007.05.25	os	černá	-	203-07.jpg
204-07	fretka	2007.05.30	zv	hnědá	bílé břicho	204-07.jpg

CREATE TABLE ztraty_verejne (evc VARCHAR (10) NOT NULL PRIMARY KEY, popis VARCHAR (60), datum DATE, vec ENUM ('k', 'p', 'kl', 'os', 'zv'), barva VARCHAR (20), spec_znak VARCHAR (30), foto VARCHAR (15), INDEX (vec));

Tabulka obsahuje všechny sloupce jako tabulka: „nalezy_verejne“ a navíc je doplněna sloupci barva, specifický znak a fotografií věci nebo zvířete. Všechny datové typy jsme si již představili u předchozích tabulek. Jedna změna tu ovšem oproti tabulce: „nalezy_verejne“ je. Jde o sloupec věc, kde je přidána značka pro označení zvířat „zv“. Je to opět z důvodu rychlejšího vyhledávání v databázi, proto je i tento sloupec indexován. Označení pro zvířata se v tabulce pro nálezy nevyskytuje z důvodu, protože zvířata nelze jednoduše skladovat někde ve skladě. Městská policie Kyjov má zařízení, do kterého je možné dočasně umístit zaběhlé psy, ale jen maximálně na dobu jednoho až tří dnů. Poté jsou převezena do útulku pro psy. Po tuto krátkou dobu je majiteli umožněno vyzvednout si svého psa přímo na Městské policii Kyjov. V opačném případě je odkázán na útulek.

Druhou tabulkou popisujícím ztráty je: „ztraty_verejne_detail“

Tabulka 6: ztraty_verejne_detail

evc	druh	Typ	cislo_ramu	vyrob-ce_znacka
200-07	pánské	Trek	xxxyyxxx	weekend
205-07	dámské	Silniční	xyxxyyxx	eska
206-07	dětské	MTB	yxxxyxxx	velamos

Syntaxe příkazu pro její vytvoření:

CREATE TABLE ztraty_verejne_detail (evc VARCHAR (10) NOT NULL PRIMARY KEY, druh ENUM ('dámské', 'pánské', 'dětské'), typ VARCHAR (20), cislo_ramu VARCHAR (20), vyrobce_znacka VARCHAR (30));

Tabulka doplňuje předchozí tabulku: „ztraty_verejne“ o některé upřesňující údaje týkající se jízdních kol. Obě dvě tabulky jsou přístupny široké veřejnosti. Je to z důvodu, aby sami uživatelé mohli být nápomocni při hledání různých věcí, nebo zvířat.

Poslední tabulkou je. „ztraty_soukrome“

Tabulka 7: ztraty_soukrome

evc	maj_jm	maj_prij	maj_obec	maj_ulice	maj_cp	maj_kon	stav

200-07	Petr	Pan	Kyjov	Husova	155	777888999	nalezeno
201-07	Tomáš	Fuk	Ježov	-	247	518555333	nenaleze-no

CREATE TABLE ztraty_soukrome (evc VARCHAR (10) NOT NULL PRIMARY KEY, maj_jm VARCHAR (15), maj_prij VARCHAR (15), maj_obec VARCHAR (20), maj_ulice VARCHAR (20), maj_cp VARCHAR (10), maj_kon VARCHAR (20), stav ENUM ('nalezeno', 'nenalezeno'));

Tabulka obsahuje informace o majiteli věci nebo zvířete, které se ztratilo. Tato data mají soukromí charakter a proto jsou určena jen oprávněným osobám. V posledním sloupci tabulky jsou indikovány aktuální stavy jednotlivých ztrát.

Tabulka: „ztraty_soukrome“ je poslední třetí tabulkou popisující nahlášené ztráty a zároveň poslední tabulkou celé databáze: „ztraty_nalezy“.

3.3 Příkaz SELECT

Aby databáze mohla fungovat a být využívána k účelu, pro který byla vytvořena, musí existovat něco, co nám uložená data zobrazí v podobě, jaké si budeme přát. Tím je v SQL příkaz: „SELECT“. Tento příkaz se používá k získávání dat z tabulek databáze výběrem řádků odpovídajících specifikovaným kritériím výběru. Existuje mnoho cest a způsobů, jak používat příkaz SELECT. Základní zápis tohoto příkazu je:

SELECT položky FROM tabulky WHERE podmínka GROUP BY typ_skupiny HAVING where_definice ORDER BY typ_řazení LIMIT limitní_kritéria;

Jak vidíme, v základním zápise příkazu SELECT je hned několik klauzulí. Ne vždy je zapotřebí využít všechny spíše naopak. I v mém případě nebude zapotřebí použít všechny klauzule v příkazu SELECT vyjmenované. Za klíčovým slovem SELECT pod označením: „položky“ mohu vypsat jednotlivé sloupce, které požaduji pro zobrazení. Místo vypisování sloupců, mohu také použít operátor, jehož zástupný symbol je: „ * “. Tento symbol zastupuje všechny sloupce ve zvolené tabulce, nebo tabulkách. Tabulky se vypisují za klauzulí FROM. Nyní mám specifikováno jak se vybírají sloupce z tabulek. Teď ukáži jak vybrat určité řádky z tabulky. Pokud totiž zadám symbol „*“ pro zápis sloupců a vyberu, z které tabulky, zobrazí se nejen všechny sloupce tabulky, ale také všechny její řádky. Chci-li tedy přistupovat jen k určité podmnožině řádků tabulky, použiji klauzuli WHERE. Klauzule WHERE specifikuje kritéria, která se mají použít pro výběr určitých řádků. Uvedu příklad:

*SELECT * FROM nalezy_verejne WHERE evc = 150.07;*

Tento příkaz mi vybere všechny sloupce z tabulky: „naxezy_verejne“, ale pouze ty řádky, které mají ve sloupci: „evc“ hodnotu: „150.07“. Jelikož je sloupec: „evc“ v tabulce: „nalezy_verejne“ zároveň primárním klíčem, vybere nám příkaz pouze jeden řádek tabulky s hodnotou: „150.07“ ve sloupci: „evc“. Tato podmínka se bude v mých definicích dotazů vyskytovat nejčastěji. Pro klauzuli WHERE existuje ještě několik dalších porovnávacích operátorů než je rovnost: „=“. Jsou to větší: „>“, menší: „<“, větší nebo rovno: „>=“, nebo také není rovno: „!=“. Operátorů je samozřejmě mnohem více. Mohu také použít několik kritérií v podmínce WHERE současně a spojit je pomocí AND a OR. Opět příklad:

*SELECT * FROM nalezy_verejne WHERE evc = 150.07 or 151.07;*

Příkaz vybere všechno jako v předchozím příkaze, jen s tím rozdílem, že vybere dva řádky, které mají ve sloupci: „evc“ hodnotu: „150.07, nebo 151.07“. Ovšem pokud existují.

Poměrně často se mi bude hodit klauzule LIMIT a ORDER BY. Tyto klauzule jsou určeny pro seřazení výsledků dotazu. Jistě by se mi nelíbilo, kdyby mi na stránce vyskočilo několik desítek řádků záznamů a navíc zpřeházené. Chci přeci uživatele zaujmout ne odradit, proto jim záznamy uspořádám do čitelné podoby. Uvedu příklad:

SELECT nv., nvd.* FROM nalezy_verejne as nv, nalezy_verejne_detail as nvd WHERE nv.evc = nvd.evc ORDER BY nv.datum DESC LIMIT 0, 15;*

Tento dotaz postupně rozeberu. Nejprve začnu druhou položkou příkazu, klauzulí FROM. Zde zadávám, které tabulky mají být použity pro dotaz a zároveň je tabulkám přiřazena zkratka tzv. ALIAS, pod kterou je tabulka definována. Tato pomůcka usnadňuje práci, která by byla při psaní zvláště dlouhých názvů tabulek. Využiji tuto pomůcku i já a budu ji používat i v dalších definicích dotazů. Takže zpět k dotazu. Chci zobrazit všechny sloupce z tabulek *nv* a *nvd*. Podmínkou WHERE vytvořím spojení tabulek na rovnost (proberu v následující kapitole). Klauzule ORDER BY mi umožní zobrazení řádků získaných z dotazu v konkrétním pořadí. Tato schopnost se hodí pro získání dobře čitelného výsledku. Jelikož chci, aby řádky byly vybrány podle datumu od nejnovějších po nejstarší záznamy, použiji klauzuli ORDER BY na sloupec datum v tabulce *nv*. Klíčové slovo DESK specifikuje řazení záznamů sestupně podle datumů. Poslední klauzulí dotazu je LIMIT. Tato klauzule se hojně používá pro vytváření webových aplikací. Určuje, které řádku výsledku dotazu mají být vypsány na výstup. Má dva parametry. První číslo určuje řádek, od kterého se má začít a druhé kolik řádků se má vrátit. Řádky jsou číslovány od nuly, tzn. že první řádek výstupu má číslo nula.

3.3.1 Spojování tabulek

Často budu potřebovat odpověď z databáze, pro kterou bude třeba použít data ze dvou a více tabulek. To se stane například v případě, že si budu chtít zobrazit popis nalezených věcí, který je v tabulce: „nalezy_verejne“, a k nim zobrazit detaily nálezu z tabulky: „nalezy_verejne_detail“. Abych mohl dát všechny informace dohromady, musím využít operace, které se říká: „spojení“, anglicky: „join“. Znamená to jednoduše, že spojím dvě a více tabulek dohromady tak, abych dodržel relace jejich dat. Ačkoli je spojování tabulek pojmově jednoduché, ve skutečnosti je to jedna z nejsložitějších částí SQL. V MySQL je implementováno několik různých typů spojení tabulek, které se užívají v rozdílných případech. Ukáži to na příkladu:

SELECT nalezy_verejne. FROM nalezy_verejne, nalezy_verejne_detail WHERE nalezy_verejne_detail.cislo_ramu = '88B88' and nalezy_verejne_detail.evc = nalezy_verejne.evc;*

V příkaze definuji zobrazení všech sloupců z tabulky: „nalezy_verejne“. Dále jsem zadal dvě tabulky: „nalezy_verejne a nalezy_verejne_detail“. Tabulky jsem oddělil čárkou, čímž jsem specifikoval jejich spojení. Tomuto typu spojení se říká: „úplné spojení (full join), nebo také kartézský součin“. Toto spojení jakoby spojí obě tabulky do jedné velké, aby každý řádek velké tabulky obsahoval všechny možné kombinace řádků obou původních tabulek. Dostanu tak tabulku, která má všechny řádky tabulky: „nalezy_verejne“ zkombinované se všemi řádky z tabulky: „nalezy_verejne_detail“. To v mnohých případech nedává příliš smysl. Většinou chci vidět jen ty řádky, které sobě odpovídají. Toho jsem dosáhl umístěním podmínky spojení v klauzuli WHERE. To je speciální typ podmíněného výrazu, který vysvětluje, které atributy tvoří vztah mezi dvěma tabulkami. Tím podmíněným výrazem je: „nalezy_verejne_dedail.evc = nalezy_verejne.evc“. Výraz říká MySQL, aby se do výsledné tabulky zahrnuly jen ty řádky tabulky: „nalezy_verejne_detail“, kde sloupec: „evc“ odpovídá sloupci: „evc“ z tabulky: „nalezy_verejne“. Přidáním spojovací podmínky do dotazu jsem změnil typ spojení na jiný, kterému se říká: „spojení na rovnost“, anglicky: „equi-join“. Uplně stejný význam, ale jednodušší zápis má spojení: „inner join“. Jde o vnitřní spojení a provede stejný výběr jako spojení na rovnost [7].

SELECT nalezy_verejne. nalezy_verejne_detail.* FROM nalezy_verejne inner join nalezy_verejne_detail on nalezy_verejne_detail.evc = nalezy_verejne.evc;*

Ovšem ani toto spojení mi neprovede výběr, který potřebuji. Spojení mi provede výběr jen těch záznamů, které se nachází zároveň v obou tabulkách. Tedy mají ve sloupci: „evc“ stejnou hodnotu. Pokud tedy budu mít v tabulce: „nalezy_verejne“ záznam, který již nebude doplněn v tabulce: „nalezy_verejne_dedail“, příkaz jej vůbec nevybere a nemůže být zobrazen uživateli. Tento problém řeší vnější spojení. Toto spojení může být levé, nebo pravé: „left join, right join“.

SELECT nalezy_verejne. nalezy_verejne_detail.* FROM nalezy_verejne_detail left join nalezy_verejne on nalezy_verejne.evc = nalezy_verejne_detail.evc;*

Příkaz je zcela totožný s předešlým, jen je upraveno spojení tabulek na: „left join“. Tím docílím toho, že mi příkaz vybere i ty záznamy, kterou jsou jen v jedné tabulce.

V databázi budu také definovat dotazy, které budou provádět výběr z více tabulek než dvou. Spojení provedu úplně stejně, protože vždy budu spojovat dvojice tabulek. Postupně tak spojím všechny tabulky. Jednotlivá spojení přitom vůbec nemusí být stejná. Mohu například dvě tabulky spojit pomocí vnitřního spojení a třetí tabulku připojit pomocí vnějšího spojení. Pro kontrolu je dobrým pravidlem, že počet spojení je o jednu menší než počet tabulek, které spojuji.

SELECT nalezy_verejne. nalezy_verejne_detail.* nalezy_soukrome_detai.* FROM nalezy_verejne inner join nalezy_verejne_detail on nalezy_verejne.evc = nalezy_verejne_detail.evc left join nalezy_soukrome_detail on nalezy_verejne.evc = nalezy_soukrome_detail.evc;*

Spojení tabulek je v relačních databázích velmi důležitým prvkem. Podcenění této operace může způsobit velké problémy a uživatele taková databáze jistě nezaujme. Proto jsem zvolil tento obsáhlejší rozbor problému: „spojování tabulek“. V práci budu používat vnější spojení.

4 PROPOJENÍ PŘES WEBOVÉ ROZHRANÍ

V této kapitole se budu zabývat propojením databáze: „ztraty_nalezy“ s webovou prezentací Městské policie Kyjov. V několika bodech nyní stručně nastíním, jak celá komunikace funguje.

1) Internetový prohlížeč vyšle požadavek http na konkrétní stránku. Například uživatel může zažádat pomocí formuláře HTML o vyhledání všech nalezených jízdních kol, které jsou uloženy v databázi: „ztraty_nalezy“. Výsledná stránka se bude jmenovat například: „ *hledej_jizdnikolo.php*“.
2) Webový server přijme žádost o soubor *hledej_jizdniholo.php*, načte soubor a předá ho PHP ke zpracování.
3) „Hnací motor“, neboli engine, PHP začne zpracovávat skript. Ve skriptu je obsažen příkaz pro připojení k databázi a k vykonání nějakého dotazu (vyhledání jízdních kol). PHP otevře připojení k serveru MySQL a odešle patřičný dotaz.
4) MySQL server přijme dotaz, zpracuje ho a odešle výsledek (seznam jízdních kol) zpět PHP.
5) PHP dokončí zpracování skriptu, což většinou zahrnuje zformátování výsledku dotazu v úhlednou stránku HTML. Ta se poté vrátí webovému serveru.
6) Webový server předá HTML zpět internetovému prohlížeči, čímž se seznam jízdních kol dostane k uživateli.

Z postupu vyřizování dotazu na databázi je vidět, že tím prostředníkem mezi webovou prezentací a databází je PHP. PHP je opravdu motor, který běží na serveru a zpracovává jednotlivé skripty.

4.1 Základy databázových dotazů z webového rozhraní

Ve skriptech přistupujících k databázi z webového rozhraní se budu držet následujících kroků:

1) Otestování a odfiltrování dat přicházejících od uživatele
2) Zřízení připojení k patřičné databázi
3) Položení dotazu
4) Zpracování výsledků
5) Zobrazení výsledků uživateli

4.1.1 Otestování a odfiltrování vstupních dat

Nejprve začnu odstraněním prázdných znaků, které by mohl uživatel nechtěně vložit na začátek a na konec termínu, který hodlá hledat. Toho docílím aplikováním funkce *trim( )* na proměnnou zastupující vložený termín. Označím si ji *$hledejtermin*. Zápis je:

$hledejtermin = trim ($hledejtermin);

Dalším krokem je kontrola, zda uživatel vůbec nějaký termín k vyhledávání zadal. Jen zdůrazním, že tuto kontrolu je nutné provádět vždy až po odstranění prázdných polí z konců proměnné *$hledejtermin*. Pokud bych tyto kroky prohodil, mohla by nastat situace, kdy by termín nebyl prázdný, ale byl by to prázdný znak a ten by byl odstraněn funkcí *trim()*. V tomto případě by nebyla vyvolána zpráva o chybném vypsání termínu. Zápis:

```
if (!$hledejtermin ||!$hledejtyp)
{
        echo „Neuvedli jste všechny informace potřebné ke hledání. Vraťte se prosím zpět a
        zkuste to znovu.“;
        exit;
}
```

V zápise jsem uvedl i novou proměnnou *$hledejtyp*, která zastupuje jednu z hodnot formulářové značky SELECT v HTML. Tento údaj bude vždy zadán, ale proč jej testuji? Důvod je jednoduchý. V budoucnu se může stát, že pro přístup do databáze může být využíváno více než jedno rozhraní. Také je dobré zkontrolovat data z bezpečnostních důvodů, protože uživatelé přistupují k databázi z různých míst.

Pokud budu zpracovávat nějaké datové vstupy ze strany uživatele, je důležité odfiltrovat případné řídící znaky. K tomu se používají dvě funkce *addslashes( ) a stripslashes( ).*

Jakmile vytvoříme pro databázi webové rozhraní, je možné, že uživatel při vkládání informací o jízdním kole vloží i některý z řídících znaků. Pro vkládání dat do databáze se používá funkce *addslashes( )* a pokud data z databáze načítám použiji funkci *stripslashes( ).* Uvedu opět zápis:

$hledejtermin = addslashes ($hledejtermin);
$hledejtermin = stripslashes ($hledejtermin);
Ještě zmíním funkci *htmlspecialchars( ).* Tato funkce se používá na zakódování znaků, které mají v HTML speciální význam. Použitím této funkce eliminuji potíže, které by mohli později vzniknout.

4.1.2 Zřízení připojení k databázi

Připojení k databázi se provádí opět pomocí funkcí. Jednou z nich je funkce *mysql_pconnect( ).* Funkce má následující prototyp:

int mysq`_pconnect ([string host [:port] [:/socketpath]],
[string user], [string password]);
Obvykle je nutné zadat hostitele, na kterém běží server MySQL, uživatelské jméno pro přihlášení a heslo zadaného uživatele. Vše je volitelné. Pokud některou položku vynechám, bude použita standardní. Funkce v případě úspěchu vrací identifikátor připojení k databázi MySQL, nebo hodnotu FALSE v případě neúspěchu.

Alternativní funkcí k funkci *mysql_pconnect( )*, je funkce *mysql_connect( ).* Rozdíl mezi nimi spočívá v tom, že *mysql_pconnect( )* vrací persistentní (trvalé) připojení k databázi.

Běžné připojení k databázi bude uzavřeno v okamžiku, kdy provádění skriptu skončí, nebo v případě, že skript zavolá funkci *mysql_close( ).* Persistentní připojení zůstává otevřené i po ukončení běhu skriptu a funkcí *mysql_close( )* se zavřít nedá.

Proč bychom ale takové připojení měli chtít? Uvědomme si, že vytvoření každého připojení má jistou režii a spotřebuje určitý čas. Pokud se zavolá funkce *mysql_pconnect( )*, automaticky ještě před pokusem o nové připojení zkontroluje, zda již nebylo otevřeno persistentní připojení. Pokud tomu tak je, dá před otevřením nového přednost připojení starému. Tím šetří čas a vytížení serveru. Počet současně otevřených připojení k MySQL je limitovaný. Smyslem tohoto omezení je, aby nové žádosti o připojení byly raději zamítnuty, než

aby došlo k zahlcení prostředků na serveru nebo případně dokonce pádu celého systému. Na druhou stranu v dnešním typickém prostředí vícevláknových serverů je čas potřebný pro vykonání funkce *mysql_connect( )* zanedbatelný.

4.1.3 Položení dotazu

Nejprve je zapotřebí vybrat databázi, se kterou budeme pracovat. V předchozí kapitole jsem používal pro výběr databáze v MySQL příkaz:

mysql>USE ztraty_nalezy;
Pokud nyní chci otevřít spojení z webového rozhraní, musím to samozřejmě udělat také. Provedu to pomocí PHP voláním funkce *mysql_select_db( )*. V mém případě bude zápis:

mysql_select_db('ztraty_nalezy');

Funkce *mysql_select_db( )* má následující prototyp:

int mysql_select_db(string database [, resource database_connection]);

Pokusí se dát do užívání databázi uvedenou jako první argument. Volitelně si můžu také zadat identifikátor připojení (v tomto případě $db), pokud ho chci konkrétně specifikovat. V případě, že identifikátor neuvedu, použije se připojení, které bylo otevřené naposled. Není-li otevřené žádné připojení, bude otevřeno standardní, stejně jako kdybychom použili funkci *mysql_connect( )*.

Dotazy se vykonávají pomocí funkce *mysql_query( )*. Ještě před použitím této funkce je dobré si jednotlivé dotazy sestavit. Uvedu příklad:

$query = " SELECT * FROM ztraty_nalezy WHERE ".$hledejtyp. " like ' % ".$hledejtermin. " % ' ";
Tímto dotazem hledám zadanou hodnotu *$hledejtermin* ve specifikované položce *$hledejtyp.* Výraz LIKE jsem použil protože hledám shodu se vzorem, nikoli na rovnost.

Teď, když mám dotaz vytvořen, mohu jej položit.

$*hledej_jizdnikolo* = *mysql_query(* $query *);*

Funkce *mysql_query( )* použiji ještě pro specifikování způsobu kódování příkazů mezi klientem a serverem. Z hlediska databáze je PHP klient. Proto musí klient posílat příkazy ve stanoveném kódování, které požaduje server. Pokud mi požadované kódování serveru nevyhovuje, můžu nastavit svoje vlastní pomocí příkazu *SET NAMES*. To provedu následovně:

mysql_query("SET NAMES ' latin 2' ;");
Tímto příkazem donutím server komunikovat s klientem kódováním *latin2*.

Funkce *mysql_query( )* má následující prototyp. Takto zadám dotaz, který se má spustit.

resource mysql_query(string query [, resource database_connection]);

Funkce vrací identifikátor výsledku (umožní tím zpracování výsledku dotazu) v případě úspěšného provedení a hodnotu *FALSE* v případě selhání. Identifikátor výsledku si uložím, abych mohl s výsledkem něco dělat. Uložil jsem ho do proměnné: „*$hledej_jizdnikolo*“.

4.1.4 Zpracování výsledků a zobrazení uživateli

Existuje mnoho funkcí, které různými způsoby umožňují přístup k výsledkům pomocí identifikátoru výsledku. Identifikátor výsledku je klíčem pro přístup k žádnému, jednomu, nebo více řádkům vrácených zadaným dotazem. Použiji funkci *mysql_num_rows ( )*. Tato funkce mi vrátí počet řádků ve výsledku dotazu. Identifikátor výsledku ji předám následujícím způsobem.

$radky_jizdnikolo = mysql_num_rows($hledej_jizdnikolo);
Tento krok je užitečný, pokud chci zpracovávat nebo zobrazit výsledky. Vím kolik řádků je a můžu jimi procházet.

```
for ($i = 0;  $i < $radky_jizdnikolo; $i ++)
{
        $zaznam = mysql_fetch_array($hledej_jizdnikolo);
}
```

V každém průchodu cyklem *for* je použita funkce *mysql_fetch_array( ),* která zpracuje každý řádek sady výsledků a vrátí ho jako asociativní pole, kde má každá položka svůj odpovídající klíč a uloží jej do proměnné *$zaznam*. Pokud ve výsledku nejsou žádné řádky, cyklus se vůbec nebude provádět.

Mohu použít i jinou techniku zpracování výsledků s použitím cyklu while. Tato metoda nezatěžuje tolik databázi, jelikož nepočítá předem všechny řádky výsledku.

```
 while ($zaznam=MySQL_Fetch_Array($hledej_jizdnikolo)) echo $zaznam["evc"]."
".$zaznam["datum"]."".$zaznam["popis"]."".$zaznam["druh"]."".$zaznam["barva"]."
".$zaznam["foto"]."<BR>\n";
```

Cyklus *while* jednoduše probíhá dokud neprojde všechny řádky výsledku. V jednom cyklu je řádek zpracován a uložen jako asociativní pole a navíc je zrovna zobrazen uživateli po jednotlivých položkách příkazem *echo* a odřádkován. Funkce *mysql_num_rows( )* zde mohu využít v podmínce *if*, ve které zjistím zda není výsledek nulový. V případě že tomu tak

je, mohu uživatele informovat o tom, že jeho výběru neodpovídá žádný záznam. Často je pro uživatele matoucí, že po zadání dotazu nedostane žádný výsledek. Proto si může myslet, že žádný takový záznam neexistuje, nebo že systém nefunguje. Použití je spíše v případech, kdy uživateli nabídnu možnost hledat záznamy dle jeho zadaných údajů. Zde se mohu domnívat, že ve výsledku výběru bude jen jeden záznam, nebo žádný. Potom kód může vypadat následovně:

```
$radky_jizdnikolo = mysql_num_rows($hledej_jizdnikolo);
if ($radky_jizdnikolo==0)
{
echo “Záznam nenalezen“;
exit;
}
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."";
 if($zaznam["foto"]!=null) echo "<a href=\"".$cesta."".$zaznam["foto"]."\
" target= \"_blank\">foto</a>";
 echo "<BR>\n";
}
```

Skript jsem ještě doplnil zobrazením výsledků pomocí cyklu *while*. Pokud proměnná *$radky_jizdnikolo* neobsahuje žádný záznam, je s tím uživatel seznámen a skript dále nepokračuje. V opačném případě jsou uživateli zobrazeny záznamy pomocí cyklu *while*. Ten je navíc doplněn podmínkou *if*, která zobrazí odkaz na fotografii záznamu, pokud existuje. Tento způsob zobrazení budu používat v případech, ve kterých budu chtít umožnit uživateli zobrazení fotografií k daným záznamům.

4.2 Vytvoření připojení k databázi

Jelikož je připojení k databázi něco, co budu ve skriptech mnohokrát používat, vytvořím si předdefinovaný soubor s konstantami. Tento soubor poté jednoduše vložím do skriptu kde jej budu potřebovat. Výhod této metody využiji jen v případech, kdy se budu přihlašovat do databáze jako uživatel s nejnižší úrovní práv označeného jako: „vsichni“. Tento uživatel nemá nadefinované přihlašovací heslo, proto nefiguruje ani v souboru, který si nazvu: „připojeni.php“.

```
<?php
if($_SERVER["SERVER_ADDR"]=="localhost")
{
  define("SQL_HOST","localhost");
```

```
  define("SQL_DBNAME","ztraty_nalezy");
  define("SQL_USERNAME","vsichni");
}
else
{
  define("SQL_HOST","mysql.dat.cz");
  define("SQL_DBNAME","ztraty_nalezy");
  define("SQL_USERNAME","vsichni");
}
?>
```

Tento soubor mi definuje přihlašovací údaje a databázi, se kterou chci pracovat. Soubor navíc obsahuje proměnnou prostředí: *$_SERVER*, pomocí níž zjistím kde právě kód běží a podle toho mi funkce: *if* vybere ty správné přihlašovací údaje. Toto provedení jsem zvolil z toho důvodu, že dané vývojové prostředí mám doma na svém počítači a reálná aplikace bude běžet na serveru poskytovatele.

Celé připojení k databázi uživatele: „vsichni“ potom bude vypadat následovně:

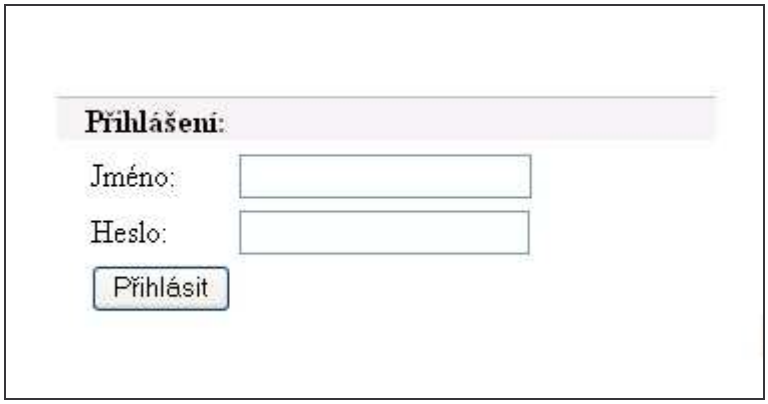
```
<?php
include("pripojeni.php");
mysql_pconnect(SQL_HOST, SQL_USERNAME) or die("Nelze se připojit k MySQL: ".mysql_error());
mysql_select_db(SQL_DBNAME) or die("Nelze vybrat databázi: ". mysql_error());
?>
```

Direktivou *include* jsem vložil do skriptu nadefinované konstanty, které jsou použity pro vytvoření trvalého připojení k databázi funkcí *mysql_pconnect( )*. Ošetřil jsem také možnost výskytu chyb. První je, kdyby se nepodařilo připojit k databázi a druhá se týká výběru databáze. V obou případech jsem použil příkaz *die*, který přeruší běh prováděného skriptu. Na tento příkaz je třeba dát si pozor. Protože pokud by tento skript byl vložen ještě do jiného skriptu, ukončí běh i tohoto rodičovského skriptu. Příkaz *die* skript ukončí, ale ještě zobrazí text následující za ní. Funkce *mysql_error( )* vypíše chybovou hlášku z MySQL.

S připojením k databázi, pod uživatelským účtem označeným jako: „straznik“, to bude trochu složitější. Tento uživatel již má oprávnění k prohlížení všech informací uložených v databázi, tedy i neveřejných, a proto je tento účet zabezpečen přihlašovacím heslem. Aby se uživatel mohl k databázi připojit pod tímto uživatelským účtem, musí se nejprve identifikovat zadáním přihlašovacího jména a hesla. To uživateli umožním vložením formuláře do stránky webové prezentace Městské policie Kyjov v sekci: „Ztráty a nálezy“. Data z formuláře bude zpracovávat skript PHP a v případě správně vložených dat, vytvoří připojení k databázi pod tímto uživatelským účtem.

Formulář bude obsahovat tři pole. Do jednoho se bude zadávat jméno, do druhého heslo a třetím se budou data odesílat. Kód tohoto formuláře je:

```
<FORM METHOD="post" ACTION="prihlaseni_straznik.php">
Jméno:<INPUT TYPE="text" NAME="jmeno">
Heslo:<INPUT  TYPE="password" NAME="heslo">
<INPUT TYPE="submit" VALUE="Přihlásit">
</FORM>
```



Obrázek 1: formulář pro přihlášení

Než vytvořím skript, který bude data z formuláře zpracovávat, chci poukázat na některá úskalí. Tím hlavním je, že protokol http je bezstavový. Znamená to, že v protokolu není nijak zabudované, jak uchovávat stav mezi dvěma transakcemi. Když uživatel požaduje nějakou stránku, a pak jinou, neposkytuje http žádnou možnost, jak bychom mohli zjistit, že oba požadavky přišli od stejného uživatele. V tomto případě by se uživatel musel přihlásit pokaždé, pokud by chtěl přejít na jinou chráněnou stránku. V takovém případě bychom zřejmě neuspěli. Možné řešení tohoto problému je v použití cookies nebo sessions. Cookies jsou relativně složité. Server je musí vygenerovat a poslat prohlížeči, který je následně posílá zase zpět na server. Sessions jsou přímo na serveru a nemusí se přeposílat tam a zpět. Není to tak úplně pravda. Opět se posílají data mezi serverem a prohlížečem, ale jen identifikační číslo sessions. Toto číslo se může vkládat přímo do URL adresy, nebo posílat pomocí cookies. Vkládání identifikačního čísla do adresy není příliš bezpečné, proto se jeví jako nejlepší možnost vzájemného využití sessions a cookies. Bohužel i tato varianta má slabinu. Pokud bude mít uživatel zakázáno přijímat cookies, nebude moci server předat prohlížeči uživatele identifikační číslo sessions a tím se uživatel nedostane na zabezpečené stránky. V mém případě to problémem nebude a proto zvolím tuto metodu. Sessions je vlastně mechanismus, který řídí PHP. Pokud PHP dostane příkaz k vytvoření

sessions, zjistí nejprve, zda již sessions neběží. Pokud ne, tak ji vytvoří, pokud ano tak se připojí. PHP ji přidělí identifikační číslo a vyhradí si někde místo pro ukládání. Od tohoto kroku si programátor může zvolit libovolné proměnné, které může registrovat do sessions. Vše je uloženo přímo na serveru a ten si pak jejich obsah pamatuje mezi stránkami. Můžeme si nakonfigurovat uložení sessions i do databáze, ale zůstanu u serveru. Sessions pak můžu kdykoli ukončit. Když tak neudělám, zruší se zavřením prohlížeče.

Nyní se vrátím k připojení k databázi pod uživatelským účtem: „straznik“. Skript, který bude vložená data do tohoto formuláře zpracovávat, jsem pojmenoval: „*prihlaseni_straznik.php*“.

```
<?php
sessions_start( );
if (isset ($HTTP_POST_VARS ["jmeno"] ) && isset ($HTTP_POST_VARS ["heslo"] );
{       //uživatel právě zadal data
$jmeno = ($HTTP_POST_VARS ["jmeno"] );
$heslo = ($HTTP_POST_VARS ["heslo"] );
$jmeno = addslashes( trim ($jmeno));
$heslo = addslashes( trim ($heslo));
mysql_connect("mysql.dat.cz", $jmeno, $heslo) or die("Špatně zadané jméno nebo heslo.
Vraťte se zpět a zkuste se přihlásit znovu);
mysql_select_db("ztraty_nalezy") or die("Nelze vybrat databázi: ". mysql_error());
sessions_register ("opravneny_uzivatel");
$_sessions ["opravneny_uzivatel"] =  $jmeno;
}
?>
```

Tímto poměrně jednoduchým skriptem připojím uživatele k databázi. Samozřejmě pokud zadá správné přihlašovací údaje. Ve skriptu je úvodem vytvořena sessions a provedena kontrola, zda uživatel nějaká data zadal. V případě že ano, jsou tato data odfiltrována od prázdných a hlavně řídících znaků. Poté jsou takto ošetřená data použita pro vytvoření připojení k databázi. Pokud nejsou data pravdivá, je skript okamžitě ukončen. V případě správnosti dat je připojení vytvořeno a skript pokračuje dál k výběru databáze. I v tomto okamžiku bude skript okamžitě ukončen nedojde-li k výběru databáze: „ztraty_nalezy“.V opačném případě je databáze vybrána a následuje zaregistrování proměnné do sessions. Hned poté je této nové proměnné přiřazeno identifikační číslo uživatele. Od tohoto okamžiku mohou být uživateli zobrazeny chráněné informace. V případě, že si uživatel chce zobrazit chráněné informace na jiné stránce, nemusí se znovu přihlašovat jako na první stránce. Tato druhá stránka obsahuje skript, který uživatele prověří automaticky sám, a je-

li uživatel řádně přihlášen, zobrazí mu informace na požadované stránce. Kód skriptu je následující:

```
<?php
sessions_start( );
if (isset ($_sessions ["opravneny_uzivatel"]))
{
echo "Jste přihlášeni jako: ".$_sessions ["opravneny_uzivatel"]." <BR />";
echo "<a href = "odhlasit.php">Odhlásit</a><BR />";
}
else
{ echo "<P>Nejste přihlášeni.</P>";
echo "<a href = "prihlaseni_straznik.php">Musíte se přihlásit</a>"; }
?>  // následují chráněné informace
```

Skript opět vytvoří sessions, pokud již neběží a zkontroluje zda sessions obsahuje registrovaného uživatele. To prověří kontrolou proměnné *$_sessions ["opravneny_uzivatel"]*. Pokud tomu tak není, oznámí uživateli že není přihlášen a odkáže ho na stránku s přihlášením. Skript obsahuje ještě část, ve které řádně přihlášenému uživateli, zobrazí jméno, pod kterým je přihlášen a nabídne možnost odhlášení. Odhlášení uživatele je provedeno automaticky při zavření internetového prohlížeče, nebo právě nabídnutou možností ve skriptu. Zde se skript odkazuje na druhý skript pojmenovaný: „odhlasit.php“.

```
<?php
sessions_start( );
$stary_uzivatel = $_sessions ["opravneny_uzivatel"];
unset ($_sessions ["opravneny_uzivatel"]);
sessions_destroy( );
?>
<HTML>
<BODY>
<H1>Odhlášení</H1>
<?php
if (! empty ($stary_uzivatel))
echo "Byli jste odhlášeni." <BR />";
else
 echo "<P>Nebyli jste přihlášeni, proto jste nemohli být odhlášeni.</P>";
?>
<a href = "index.php">Zpět na domovskou stránku</a>
</BODY>
</HTML>
```

Na začátku skriptu je opět sessions vytvořena v případě že již neběžela. Následuje uložení hodnoty z proměnné *$_sessions ["opravneny_uzivatel"]* do nově vytvořené proměnné *$stary_uzivatel*. Poté je zrušena registrace proměnné *$_sessions ["opravneny_uzivatel"]* a zlikvidována sessions. Poté obeznámím uživatele o tom, že byl odhlášen, nebo že odhlášen

být nemohl, protože vůbec přihlášen nebyl. To pro případ, že se uživatel na tuto stránku dostal nějak jinak.

Samotné přihlášení uživatele do systému je tímto vyřešeno. Je tu ale ještě jedna důležitá věc. Přenos informací od uživatele k serveru není nijak zabezpečen, proto může kdokoli tuto komunikaci odposlouchávat a přijít tak k přihlašovacím údajům oprávněného uživatele. Proto použiji protokol SSL (Secure Socket Layer). Protokol byl vyvinut firmou Netscape pro zajištění bezpečné komunikace mezi webovými servery a webovými prohlížeči. Protokol SSL je všeobecně podporován. Většina webových serverů má SSL přímo implementován, nebo jej přijímá jako doplňkový modul. Webové prohlížeče mají funkci SSL zabudovanou téměř všechny. Webový prohlížeč se nejdříve spojí protokolem http s bezpečným webovým serverem, dojde k úvodní komunikaci, kdy se SSL „domluví“ s prohlížečem na shodě konkrétních věcí, jako jsou prokázání totožnosti a šifrování. Celá procedura navázání spojení se může protáhnout, protože při ní dochází k řadě úkonů. Jde především o vygenerování kvalitních náhodných dat, dešifrování digitálních certifikátů, vygenerování klíčů a použití kryptografie veřejných klíčů. Výsledky jsou ukládány do vyrovnávací paměti, takže pokud si prohlížeč se serverem vyměňují více bezpečnostních zpráv, není potřeba provádět úvodní domluvu opětovně. Nyní již může dojít k bezpečnému přenosu jednotlivých dat. Data jsou ještě před odesláním upravena a zašifrována. Komunikace pomocí protokolu SSL je bezpečná, nicméně dosti časově i výkonově náročná. Proto tuto zabezpečenou komunikaci prohlížeče a serveru použiji jen při přenose přihlašovacích údajů.

4.3 Vytvoření dotazů z webového rozhraní

O dotazech na databázi MySQL jsem již mluvil několikrát. Nyní se budu zabývat dotazy, které jsou zadávány uživateli přímo z webového rozhraní. MySQL dotazy sami o sobě z webového rozhraní fungovat nebudou. Zprostředkovatelem se v tomto kroku stává PHP. Skripty PHP zprostředkují přenos mezi webovým rozhraním a databází MySQL uloženou na serveru. Samotné MySQL dotazy jsou do těchto PHP skriptů zakomponovány.

Nejprve začnu vytvořením dotazů pro uživatele, kteří se budou připojovat do databáze pod uživatelským účtem označeným jako: „vsichni“, a budou chtít zobrazit záznamy o nalezených věcech. První dotaz bude implicitní, který se provede ještě dříve než si to vůbec uživatel stačí uvědomit. Tento dotaz se začne provádět v okamžiku, kdy uživatel klikne na

stránku webové prezentace Městské policie Kyjov v sekci: „Ztráty a nálezy“. Automaticky se zobrazí výsledek dotazu se všemi záznamy o nálezech, seřazenými podle datumu od nejnovějšího po nejstarší záznam. Záznamy nebudou zobrazeny všechny zároveň, nýbrž po patnácti což nám umožní klauzule LIMIT. V PHP terminologii se tomuto kroku říká stránkování. Kód tohoto skriptu je:

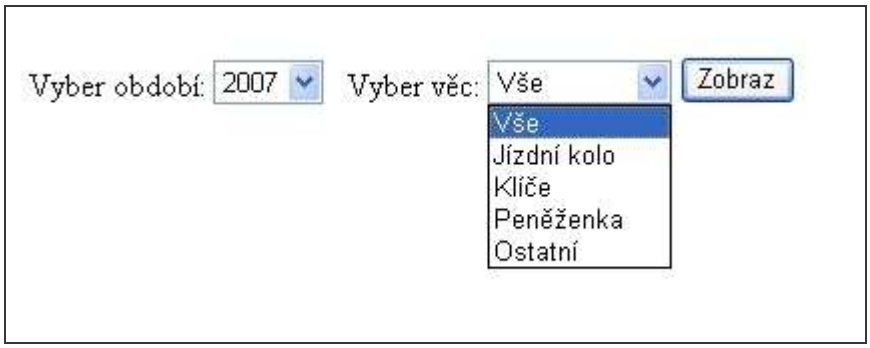
```
<?php
define("RADKU",15);
include("pripojeni.php");
mysql_pconnect(SQL_HOST, SQL_USERNAME) or die("Nelze se připojit k MySQL: ".mysql_error());
mysql_select_db(SQL_DBNAME) or die("Nelze vybrat databázi: ". mysql_error());
if (!isset ($_GET["celkem"])) //zde zjistím kolik záznamů bude celkem
 {
  $vysledek=mysql_query("select count(*) as pocet from nalezy_verejne");
  $zaznam=mysql_fetch_array($vysledek);
  $celkem=$zaznam["pocet"];
 }
 else { $celkem=$_GET["celkem"]; }
 if ($celkem>RADKU)
 { if (!isset($_GET["od"])) $od=1; else $od=$_GET["od"];
mysql_query("SET NAMES ' latin 2' ;" );
$vysledek=mysql_query("SELECT nv.evc, datum, popis, nvd.druh, barva, foto_ostatni FROM nalezy_verejne as nv left join nalezy_verejne_detail as nvd on nv.evc = nvd.evc ORDER BY nv.datum DESC "."LIMIT ".($od-1).", ".RADKU);
// zde je vložený MySQL dotaz, záznamy se uloží do proměnné $vysledek
 echo"Záznamů:".$od."-";
 echo (($od+RADKU-1)<=$celkem)?($od+RADKU-1):$celkem;
 echo " z celkem $celkem   ";
 //začátek - vytvoř odkaz pouze pokud nejsme na začátku
 if ($od==1) echo "Začátek | ";
else echo "<a href=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=1\">
Začátek</a> | ";
 //zpět - vytvoř odkaz pouze pokud nejsme v prvních RADKU
 if ($od<RADKU) echo "Předchozí | ";
 else echo "<a href=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=".($od-RADKU)."\">Předchozí</a> | ";
 //další – vytvoř odkaz pouze pokud nejsme v posledních RADKU
 if ($od+RADKU>$celkem) echo "Následující | ";
 else echo "<a href=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=".($od+RADKU)."\">Následující</a> | ";
 //poslední - to je posledních (zbytek po dělení RADKU) záznamů
 if ($od>$celkem-RADKU) echo "Konec <BR>";
 else echo "<ahref=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=
".($celkem-$celkem%RADKU+1)."\">Konec</a><BR>";
 }
$cesta="../Obrazky/ztraty_nalezy/"; // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
```

```
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."";
 if($zaznam["foto_ostatni"]!=null)echo "<a href=\"".$cesta."".$zaznam["foto_ostatni"]."\
" target= \"_blank\">foto</a>";
 echo "<BR>\n";
}
?>
```

Tímto poměrně dlouhým skriptem docílím toho, aby se mi záznamy zobrazovali jen po patnácti najednou a na následující či předchozí záznamy byl vytvořen odkaz. Tento soubor si nazvu: „*Ndotaz_implicitni_vsichni.php*“ a budu jej používat i dále v následujících dotazech. Bude zapotřebí vždy jej trochu poupravit. Jednak samotný MySQL dotaz, který do skriptů budu vkládat a jiné přihlašovací údaje, půjde-li o uživatele s větším oprávněním.

Budu pokračovat dál ve vytváření dotazů pro uživatelský účet označený jako: „vsichni“. Po zobrazení webové stránky s předešlým skriptem bude uživateli nabídnuta možnost specifikovat výběr záznamů podle časového období (rok) a zároveň podle druhu věci, kterou chce zobrazit (jízdní kolo, klíče, peněženka a ostatní). Tuto možnost uživateli nabídnu opět pomocí formuláře.

```
<FORM METHOD=“post“ ACTION="Ndotaz_ObdobiVec_vsichni.php">
Vyber období:
<SELECT NAME="obdobi">
        <OPTION VALUE="">Vše
        <OPTION VALUE="2007">2007
        <OPTION VALUE="2006">2006
</SELECT>
Vyber věc:
<SELECT NAME="vec">
        <OPTION VALUE="">Vše
        <OPTION VALUE="k">Jízdní kolo
        <OPTION VALUE="kl">Klíče
        <OPTION VALUE="p">Peněženka
        <OPTION VALUE="os">Ostatní
</SELECT>
<INPUT TYPE=“submit“ VALUE="Zobraz">
</FORM>
```



Obrázek 2: formulář pro specifikaci období a věci

Podobný formulář budu používat častěji jen bude doplněn o jiné volby a nebo jiné druhy formulářových polí. Vždy ale půjde o to, dát uživateli možnost vložit určitá data, na základě kterých se poté provede příslušný dotaz na databázi. Skript, který bude zpracovávat data z výše uvedeného formuláře se jmenuje: „*Ndotaz_ObdobiVec_vsichni.php*“ a jeho kód je:

```
$vec = trim ($vec); $vec = addslashes ($vec);
$obdobi = trim ($obdobi); $obdobi = addslashes ($obdobi);
$vysledek=mysql_query("SELECT nv.evc, datum, popis, nvd.druh, barva, foto_ostatni
FROM nalezy_verejne as nv left join nalezy_verejne_detail as nvd on nv.evc = nvd.evc
WHERE nv.vec=' ".$vec. " ' and nv.YEAR(datum)=' ".$obdobi." ' ORDER BY nv.datum
DESC "."LIMIT ".($od-1).", ".RADKU);
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
```

Samozřejmě tento kód není celý. Je zobrazena jen ta část, která musela být upravena oproti dříve uvedenému skriptu: „*Ndotaz_implicitní_vsichni.php*“. Jedná se o úpravu samotného dotazu, kdy do podmínky WHERE byly přidány položky, které způsobí zobrazení požadovaných záznamů. Vše ostatní zůstává zachované jako v původním skriptu.

Posledním dotazem, kterým bude moci disponovat uživatel přihlášený pod uživatelským účtem označeným jako: „vsichni“, bude možnost vyhledat jízdní kolo podle uživatelem zadaného čísla rámu. Pokud uživatel v předchozí nabídce zvolil pro vyhledávání položku jízdní kolo, bude mu na následné stránce se zobrazenými záznamy jízdních kol nabídnuta možnost, hledání jízdního kola přímo podle čísla jeho rámu. Do připraveného pole uživatel zadá číslo rámu jízdního kola, které hledá a MySQL dotaz pomocí skriptu PHP prohledá databázi, jestli v ní není takový záznam uložen. Formulář bude mít jen pole pro zadání čísla rámu a druhé pole pro odeslání informací. Bude použita proměnná: „*$cislo*“, která bude odpovídat hodnotě zadané do formulářového pole s názvem: „*cislo*“. Skript který bude data zpracovávat bude následující:

```
<?php
$cislo = trim ($cislo); $cislo = addslashes ($cislo);
```

```
if ( empty ($cislo)) echo "Nezadali jste žádnou hodnotu"; exit;
 endif;  //ošetřil jsem vstupní data od uživatele
include("pripojeni.php");
mysql_pconnect(SQL_HOST, SQL_USERNAME) or die("Nelze se připojit k MySQL: ".mysql_error());
mysql_select_db(SQL_DBNAME) or die("Nelze vybrat databázi: ". mysql_error());
mysql_query("SET NAMES ' latin 2' ;" );
$vysledek=mysql_query("SELECT nv.evc, datum, popis, nvd.druh, barva, cislo_ramu FROM nalezy_verejne as nv left join nalezy_verejne_detail as nvd on nv.evc = nvd.evc WHERE nvd.cislo_ramu=' ".$cislo." ';
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
while ($zaznam = mysql_fetch_array($vysledek)) echo $zaznam["evc"]."
".$zaznam["datum"]."".$zaznam["popis"]."".$zaznam["druh"]."
".$zaznam["barva"]."".$zaznam["cislo_ramu"];
?>
```

Soubor s tímto skriptem si pojmenuji: „*Ndotaz_cislo_vsichni.php*“. Všechny dosud vytvořené dotazy se týkaly hledání informací v tabulkách se záznamy o nalezených věcech. Nyní se podíváme na dotazy, které budou prohledávat tabulky se záznamy o nahlášených ztrátách. Opět se budu věnovat uživatelskému účtu: „vsichni“. Implicitní dotaz se bude provádět, pokud uživatel klikne na stránku: „Ztráty“, webové prezentace Městské policie Kyjov v sekci: „Ztráty a nálezy“. Kód tohoto skriptu bude téměř totožný se skriptem použitým na záznamy s nálezy pod názvem: „*Ndotaz_implicitni_vsichni.php*“. Rozdíl v kódu je:

```
$vysledek=mysql_query("SELECT zv.evc, datum, popis, barva, foto, spec_znak zvd.druh, typ, cislo_ramu, vyrobce_znacka FROM ztraty_verejne as zv left join ztraty_verejne_detail as zvd on  zv.evc = zvd.evc ORDER BY zv.datum DESC "."LIMIT ".($od-1).", ".RADKU);
// …….. pokračování kódu
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]."";
 if($zaznam["foto "]!=null)echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
 echo "<BR>\n";
}
?>
```

Vypsal jsem jen odlišné části skriptu. Soubor si pojmenuji: „*Zdotaz_implicitni_vsichni.php*“. Po zobrazení webové stránky s předešlým skriptem bude opět uživateli nabídnuta možnost specifikovat výběr záznamů podle časového období (rok) a zároveň podle druhu věci, kterou chce zobrazit (jízdní kolo, klíče, peněženka, ostatní a

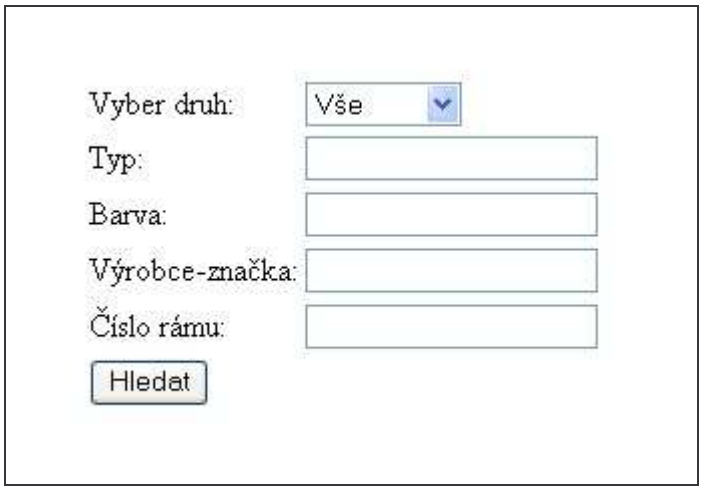
zvíře). Tyto možnosti se opakují jako při zobrazení nálezů. Doplněna je jen položka zvíře, která v nálezech nefigurovala. Formulář pro zadávání hodnot je úplně totožný jako byl na stránce nálezů, jen je rozšířen o políčko: „Zvíře“ s hodnotou VALUE="zv". Skript zpracovávající data z tohoto formuláře je také téměř stejný jako skript: „*Ndotaz_ObdobiVec_vsichni.php*“. Liší se jen v syntaxi MySQL příkazu SELECT a jinými položkami pro zobrazení na konci skriptu.

```
$vysledek=mysql_query("SELECT zv.evc, datum, popis, barva, foto, spec_znak zvd.druh,
typ, cislo_ramu, vyrobce_znacka FROM ztraty_verejne as zv left join  ztra-
ty_verejne_detail as zvd on zv.evc = zvd.evc WHERE zv.vec=' ".$vec. " ' and
zv.YEAR(datum)=' ".$obdobi." ' ORDER BY zv.datum DESC "."LIMIT ".($od-1).",
".RADKU);   // ……. pokračování kódu
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]."";
 if($zaznam["foto "]!=null)echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
 echo "<BR>\n";}
?>
```

Soubor s tímto skriptem si pojmenuji: „*Zdotaz_ObdobiVec_vsichni.php*“.

V případě, že si uživatel v předchozí nabídce zvolil položku jízdní kolo, bude mu na další stránce nabídnuta možnost specifikovat svůj dotaz. Opět použiji formulář s kódem:

```
<FORM METHOD=“post“ ACTION="Zdotaz_kolo_vsichni.php">
Vyber druh:
<SELECT NAME="druh">
        <OPTION VALUE="">Vše
        <OPTION VALUE="dámské">Dámské
        <OPTION VALUE="pánské">Pánské
        <OPTION VALUE="dětské">Dětské
</SELECT>
Typ:<INPUT TYPE=“text“ NAME="typ">
Barva:<INPUT  TYPE=“text“ NAME="barva">
Výrobce-značka:<INPUT TYPE=“tet“ NAME="vyrobce">
Číslo rámu:<INPUT TYPE=“tet“ NAME="cislo">
<INPUT TYPE=“submit“ VALUE="Hledat">
</FORM>
```



Obrázek 3: formulář pro specifikaci údajů jízdních kol

Formulář nabízí poměrně dost položek, pomocí kterých může uživatel svůj dotaz specifikovat. Skript který bude vložená data zpracovávat se jmenuje: „*Zdotaz_kolo_vsichni.php*“ a jeho kód je následující:

```
$druh = trim ($druh); $druh = addslashes ($druh);
$typ = trim ($typ); $typ = addslashes ($typ);
$barva = trim ($barva); $barva = addslashes ($barva);
$vyrobce = trim ($vyrobce); $vyrobce = addslashes ($vyrobce);
$cislo = trim ($cislo); $cislo = addslashes ($cislo);
$vysledek=mysql_query("SELECT zv.evc, datum, popis, barva, foto, spec_znak zvd.druh,
typ, cislo_ramu, vyrobce_znacka FROM ztraty_verejne as zv left join ztraty_verejne_detail
as zvd on zv.evc = zvd.evc WHERE zvd.druh=' ".$druh. " ', zvd.typ like ' % ".$typ."% ',
zvd.vyrobce_znacka like  ' % ".$vyrobce."% ', zvd.cislo_ramu=' ".$cislo."', zv.barva like '
% ".$barva."% ' ORDER BY zv.datum DESC "."LIMIT ".($od-1).", ".RADKU);
// ……. pokračování kódu
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]."";
 if($zaznam["foto "]!=null)echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
 echo "<BR>\n";
}
?>
```

Opět jsem uvedl jen odlišné části skriptu oproti původnímu skriptu: „*Ndotaz_implicitni_vsichni*“. Tímto posledním skriptem jsem skončil v tvorbě dotazů pro uživatelský účet pojmenovaný jako: „vsichni“.

V následujícím textu si vytvořím dotazy pro uživatele, kteří budou přihlášeni pod uživatelským účtem označeným jako: „straznik“. Dotazy se budou v mnohém podobat, proto budu uvádět jen ty části skriptů, které se budou lišit.

Nejdříve začnu dotazy na tabulky se záznamy o nalezených věcech. První dotaz bude implicitní. Stránka se zobrazením záznamů tohoto dotazu bude načtena ihned po přihlášení uživatele k databázi, pod uživatelským účtem: „straznik“. Kód skriptu s dotazem je následující:

```
<?php
sessions_start( );
if (isset ($HTTP_POST_VARS ["jmeno"] ) && isset ($HTTP_POST_VARS ["heslo"] );
{       //uživatel právě zadal data
$jmeno = ($HTTP_POST_VARS ["jmeno"] );
$heslo = ($HTTP_POST_VARS ["heslo"] );
$jmeno = addslashes( trim ($jmeno));
$heslo = addslashes( trim ($heslo));
mysql_connect("mysql.dat.cz", $jmeno, $heslo) or die("Špatně zadané jméno nebo heslo.
Vraťte se zpět a zkuste se přihlásit znovu);
mysql_select_db("ztraty_nalezy") or die("Nelze vybrat databázi: ". mysql_error());
sessions_register ("opravneny_uzivatel");
$_sessions ["opravneny_uzivatel"] =  $jmeno;
}       // uživatel se přihlásil

if (!isset ($_GET["celkem"]))   //zde zjistím kolik záznamů bude celkem
 {
   $vysledek=mysql_query("select count(*) as pocet from nalezy_verejne");
   $zaznam=mysql_fetch_array($vysledek);
   $celkem=$zaznam["pocet"];
 }
 else  { $celkem=$_GET["celkem"];  }
 if ($celkem>RADKU)
 { if (!isset($_GET["od"])) $od=1; else $od=$_GET["od"];
mysql_query("SET NAMES ' latin 2' ;" );
$vysledek=mysql_query("SELECT nv.evc, datum, popis nvd.druh, barva, foto_ostatni, cislo_ramu ns.typ, vyrobce_znacka, foto nsd.misto_nalezu, nalez_jm, nalez_prij, nalez_kon,
spec_znak, stav FROM nalezy_verejne as nv left join nalezy_verejne_detail as nvd on
nv.evc = nvd.evc left join nalezy_soukrome as ns on nv.evc = ns.evc left join nalezy_soukrome_detail as nsd on nv.evc = nsd.evc ORDER BY nv.datum DESC "."LIMIT
".($od-1).", ".RADKU);
 echo"Záznamů:".$od."-";
 echo (($od+RADKU-1)<=$celkem)?($od+RADKU-1):$celkem;
 echo " z celkem $celkem   ";
 //začátek - vytvoř odkaz pouze pokud nejsme na začátku
 if ($od==1) echo "Začátek | ";
else echo "<a href=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=1\">
Začátek</a> | ";
 //zpět - vytvoř odkaz pouze pokud nejsme v prvních RADKU
```

```
 if ($od<RADKU) echo "Předchozí | ";
 else echo "<a href=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=".($od-
RADKU)."\">Předchozí</a> | ";
 //další – vytvoř odkaz pouze pokud nejsme v posledních RADKU
 if ($od+RADKU>$celkem) echo "Následující | ";
 else echo "<a href=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=".($od+
RADKU)."\">Následující</a> | ";
 //poslední - to je posledních (zbytek po dělení RADKU) záznamů
 if ($od>$celkem-RADKU) echo "Konec <BR>";
 else echo "<ahref=\"".$_SERVER["PHP_SELF"]."?celkem=$celkem&amp; od=
".($celkem-$celkem%RADKU+1)."\">Konec</a><BR>";
  }
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]."".$zaznam["stav"]."
".$zaznam["místo_nalezu"]."".$zaznam["nalez_jm"]."".$zaznam["nalez_prij"]."
".$zaznam["nalez_kon"]."";
 if($zaznam["foto "]!=null)echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
if($zaznam["foto_ostatni "]!=null)echo"<a href=\"".$cesta."".$zaznam["foto_ostatni "]."\
" target= \"_blank\">foto</a>";
 echo "<BR>\n";
}
?>
```

Skript nám automaticky zobrazí všechny položky, ke kterým má uživatel oprávnění přistupovat. Záznamy nebudou zobrazeny všechny zároveň, ale pouze po patnácti na jedné straně. Na zbývající bude vytvořen odkaz. Soubor pojmenuji: „*Ndotaz_imlicitni_straznik.php*“. Následující dotaz umožní uživateli zobrazit záznamy, které si může zvolit dle časového období (rok) a nebo druhu věci. Opět je použit formulář, který je totožný s formulářem jako u uživatelského účtu: „vsichni“. Skript pro zpracování je:

```
$vysledek=mysql_query("SELECT nv.evc, datum, popis nvd.druh, barva, foto_ostatni, cis-
lo_ramu ns.typ, vyrobce_znacka, foto nsd.misto_nalezu, nalez_jm, nalez_prij, nalez_kon,
spec_znak, stav FROM FROM nalezy_verejne as nv left join nalezy_verejne_detail as nvd
on nv.evc = nvd.evc left join nalezy_soukrome as ns on nv.evc = ns.evc left join nale-
zy_soukrome_detail as nsd on nv.evc = nsd.evc WHERE nv.vec=' ".$vec. " ',
nv.YEAR(datum)=' ".$obdobi." ' ORDER BY nv.datum DESC "."LIMIT ".($od-1).",
".RADKU);
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
```

Jen tato část skriptu je odlišná, zbývající část skriptu je naprosto identická se skriptem: „*Ndotaz_implicitní_straznik.php*“. Tento nový soubor, uvedený výše, si pojmenuji: „*Ndotaz_ObdobiVec_straznik.php*“.

Pokud si uživatel zvolí položku jízdní kolo, bude mu na další stránce nabídnuta možnost, specifikovat tento druh věci podrobněji. Využiji již dříve uvedený formulář, který byl použit pro uživatelský účet: „vsichni“, při podrobném hledání jízdních kol nahlášených jako ztráta. Skript pro zpracování těchto dat je:

```
$druh = trim ($druh); $druh = addslashes ($druh);
$typ = trim ($typ); $typ = addslashes ($typ);
$barva = trim ($barva); $barva = addslashes ($barva);
$vyrobce = trim ($vyrobce); $vyrobce = addslashes ($vyrobce);
$cislo = trim ($cislo); $cislo = addslashes ($cislo);
$vysledek=mysql_query("SELECT nv.evc, datum, popis nvd.druh, barva, foto_ostatni, cislo_ramu ns.typ, vyrobce_znacka, foto nsd.misto_nalezu, nalez_jm, nalez_prij, nalez_kon, spec_znak, stav FROM FROM nalezy_verejne as nv left join nalezy_verejne_detail as nvd on nv.evc = nvd.evc left join nalezy_soukrome as ns on nv.evc = ns.evc left join nalezy_soukrome_detail as nsd on nv.evc = nsd.evc WHERE nvd.druh=' ".$druh. " ', ns.typ like ' % ".$typ."% ', ns.vyrobce_znacka like  ' % ".$vyrobce."% ', nvd.cislo_ramu=' ".$cislo."', nvd.barva like ' % ".$barva."% ' ORDER BY nv.datum DESC "."LIMIT ".($od-1).", ".RADKU);
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
```

Zbývající část skriptu je opět totožná se skriptem: „*Ndotaz_implicitni_straznik.php*“. Vytvořený soubor pojmenuji: „*Ndotaz_kolo_straznik.php*“. Tím jsem vytvořil dotazy na databázi se záznamy o nálezech.

Posledními dotazy budou opět dotazy pro uživatelský účet: „straznik“, ale pro zobrazení záznamů o nahlášených ztrátách. První bude jako vždy implicitní dotaz, který se provede ihned po přihlášení uživatele pod uživatelským účtem: „straznik“.

```
$vysledek=mysql_query("SELECT zv.evc, datum, popis, spec_znak,  barva, foto zvd.cislo_ramu, typ, vyrobce_znacka, druh zs.stav, maj_jm, maj_prij, maj_obec, maj_ulice, maj_cp, maj_kon FROM ztraty_verejne as zv left join ztraty_verejne_detail as zvd on zv.evc = zvd.evc left join  ztraty_soukrome as zs on zv.evc = zs.evc ORDER BY zv.datum DESC "."LIMIT ".($od-1).", ".RADKU);
// kód zde pokračuje
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))

 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]."".$zaznam["stav"]."
```

```
".$zaznam["maj_jm"]."".$zaznam["maj_prij"]."".$zaznam["maj_obec"]."".$zaznam
["maj_ulice"]."".$zaznam["maj_cp"]."".$zaznam["maj_kon"]."";
 if($zaznam["foto "]!=null) echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
echo "<BR>\n";
}
```

Uvedl jsem jen tu část kódu, která je odlišná od skriptu: „*Ndotaz_implicitni_straznik.php*“. Tento nový soubor si pojmenuji: „*Zdotaz_implicitni_straznik.php*“.

V dalším dotazu si uživatel může vybrat časové období (rok) a druh věci nebo zvíře. Formulář byl už také v práci uveden, při zadávání hodnot na stránce nálezů uživatelského účtu: „vsichni“. Jen je rozšířen o políčko: „Zvíře“ s hodnotou VALUE="zv", jinak je zcela totožný. Skript pro zpracování údajů je:

```
$vec = trim ($vec); $vec = addslashes ($vec);
$obdobi = trim ($obdobi); $obdobi = addslashes ($obdobi);
$vysledek=mysql_query("SELECT zv.evc, datum, popis, spec_znak, barva, foto
zvd.cislo_ramu, typ, vyrobce_znacka, druh zs.stav, maj_jm, maj_prij, maj_obec, maj_ulice,
maj_cp, maj_kon FROM ztraty_verejne as zv left join ztraty_verejne_detail as zvd on
zv.evc = zvd.evc left join ztraty_soukrome as zs on zv.evc = zs.evc WHERE zv.vec='
".$vec. " ', zv.YEAR(datum)=' ".$obdobi." ' ORDER BY zv.datum DESC ."LIMIT ".($od-
1)."
```

, ".RADKU);

```
// kód zde pokračuje
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]""]."".$zaznam["stav"]."
".$zaznam["maj_jm"]."".$zaznam["maj_prij"]."".$zaznam["maj_obec"]."".$zaznam
["maj_ulice"]."".$zaznam["maj_cp"]."".$zaznam["maj_kon"]."";
 if($zaznam["foto "]!=null) echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
echo "<BR>\n";
}
```

Zbývající část skriptu je totožná se skriptem: „*Ndotaz_implicitni_straznik.php*“. Nově vytvořený soubor pojmenuji: „*Zdotaz_ObdobiVec_straznik.php*“.

Předposledním dotazem je možnost hledání jízdního kola. Pokud uživatel v předchozí nabídce zvolil tuto položku, bude mu nabídnuta možnost specifikovat svůj dotaz. Formulář je stejný jako byl u uživatelského účtu: „vsichni“ při hledání totožné položky. Skript pro zpracování je ale trochu odlišný.

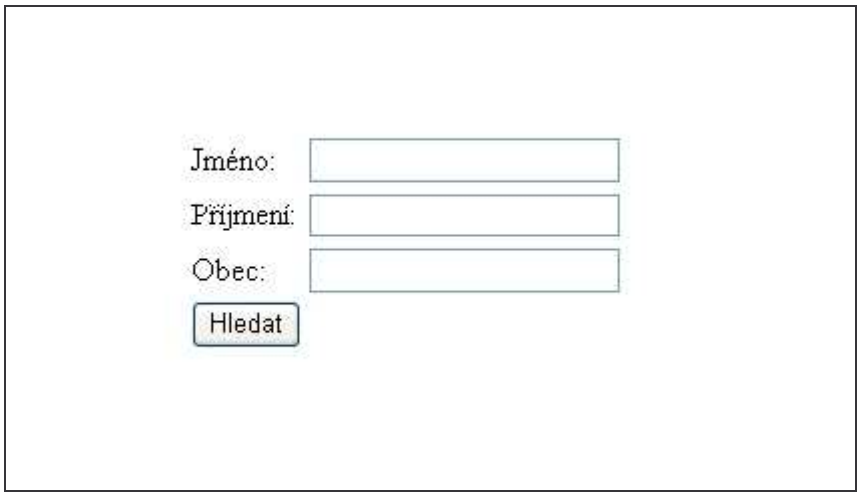
```
$druh = trim ($druh); $druh = addslashes ($druh);
```

```
$typ = trim ($typ); $typ = addslashes ($typ);
$barva = trim ($barva); $barva = addslashes ($barva);
$vyrobce = trim ($vyrobce); $vyrobce = addslashes ($vyrobce);
$cislo = trim ($cislo); $cislo = addslashes ($cislo);
$vysledek=mysql_query("SELECT zv.evc, datum, popis, spec_znak, barva, foto
zvd.cislo_ramu, typ, vyrobce_znacka, druh zs.stav, maj_jm, maj_prij, maj_obec, maj_ulice,
maj_cp, maj_kon FROM ztraty_verejne as zv left join ztraty_verejne_detail as zvd on
zv.evc = zvd.evc left join  ztraty_soukrome as zs on zv.evc = zs.evc  WHERE zvd.druh='
".$druh. " ', zvd.typ like ' % ".$typ."% ',  zvd.vyrobce_znacka like  ' % ".$vyrobce."% ',
zvd.cislo_ramu=' ".$cislo."', zv.barva like ' % ".$barva."% ' ORDER BY zv.datum DESC
"."LIMIT ".($od-1).", ".RADKU);
// kód zde pokračuje
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 { echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]."".$zaznam["stav"]."
".$zaznam["maj_jm"]."".$zaznam["maj_prij"]."".$zaznam["maj_obec"]."".$zaznam
["maj_ulice"]."".$zaznam["maj_cp"]."".$zaznam["maj_kon"]."";
 if($zaznam["foto "]!=null) echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
echo "<BR>\n";}
```

Tato část skriptu je odlišná oproti skriptu: „Ndotaz_implicitni_straznik.php“. Nově vytvořený soubor pojmenuji: „Zdotaz_kolo_straznik.php“.

Konečně poslední dotaz poskytne uživateli možnost, vyhledat záznamy podle majitele ztracené věci či zvířete. Formulář bude běžné konstrukce se třemi poli označenými jako: „jmeno, prijmeni, obec“. Kód tohoto formuláře je:

```
<FORM METHOD=“post“ ACTION="majitel.php">
Jméno:<INPUT TYPE=“text“ NAME="jmeno">
Heslo:<INPUT  TYPE=“text“ NAME="prijmeni">
Obec:<INPUT  TYPE=“text“ NAME="obec">
<INPUT TYPE=“submit“ VALUE="Hledat">
</FORM>
```



Obrázek 4: formulář pro vyhledávání záznamů podle majitele

Potom skript zpracovávající tyto údaje bude vypadat následovně:

```
$jmeno = trim ($jmeno); $ jmeno = addslashes ($jmeno);
$prijmeni = trim ($prijmeni); $ prijmeni = addslashes ($prijmeni);
$obec = trim ($obec); $obec = addslashes ($obec);
$vysledek=mysql_query("SELECT zv.evc, datum, popis, spec_znak, barva, foto
zvd.cislo_ramu, typ, vyrobce_znacka, druh zs.stav, maj_jm, maj_prij, maj_obec, maj_ulice,
maj_cp, maj_kon FROM ztraty_verejne as zv left join ztraty_verejne_detail as zvd on
zv.evc = zvd.evc left join ztraty_soukrome as zs on zv.evc = zs.evc WHERE zs.maj_jm like
' % ".$jmeno. " % ', zs.maj_prij like ' % ".$prijmeni."% ', zs.maj_obec like ' %
".$obec."% ' ORDER BY zv.datum DESC "."LIMIT ".($od-1).", ".RADKU);
// kód zde pokračuje
if (mysql_num_rows($vysledek)==0) echo "Nebyl nalezen žádný záznam"; exit;
 endif; //pokud nejsou žádné záznamy skript je ukončen
$cesta="../Obrazky/ztraty_nalezy/";  // cesta kde jsou uloženy fotografie
while ($zaznam=MySQL_Fetch_Array($vysledek))
 {
 echo"<B>".$zaznam["evc"]."</B>".$zaznam["datum"]."".$zaznam["popis"]."
".$zaznam["druh"]."".$zaznam["barva"]."".$zaznam["spec_znak"]."".$zaznam["typ"]."
".$zaznam["vyrobce_znacka"]."".$zaznam["cislo_ramu"]."".$zaznam["stav"]."
".$zaznam["maj_jm"]."".$zaznam["maj_prij"]."".$zaznam["maj_obec"]."".$zaznam
["maj_ulice"]."".$zaznam["maj_cp"]."".$zaznam["maj_kon"]."";
 if($zaznam["foto "]!=null) echo "<a href=\"".$cesta."".$zaznam["foto "]."\
" target= \"_blank\">foto</a>";
echo "<BR>\n";
}
```

Soubor pojmenuji: „*Zdotaz_majitel_straznik.php*“. Zbývající část skriptu je taktéž totožná se skriptem: „*Ndotaz_implicitni_straznik.php*“.

Pro třetí uživatelský účet označený jako: „spravce“, dotazy na databázi vytvářet nebudu. Tento účet je určen pro člověka, který bude databázi aktualizovat. Má k tomu přidělena oprávnění a jen on bude moci data vkládat, odstraňovat, atd.. Při práci bude využívat ná-

stroj PHPmyAdmin. Tento nástroj je určen k administraci v MySQL. Bylo by proto zbytečné, vytvářet nové dotazy a příkazy na databázi, když takový nástroj existuje.

4.4 Formulář pro hlášení ztrát

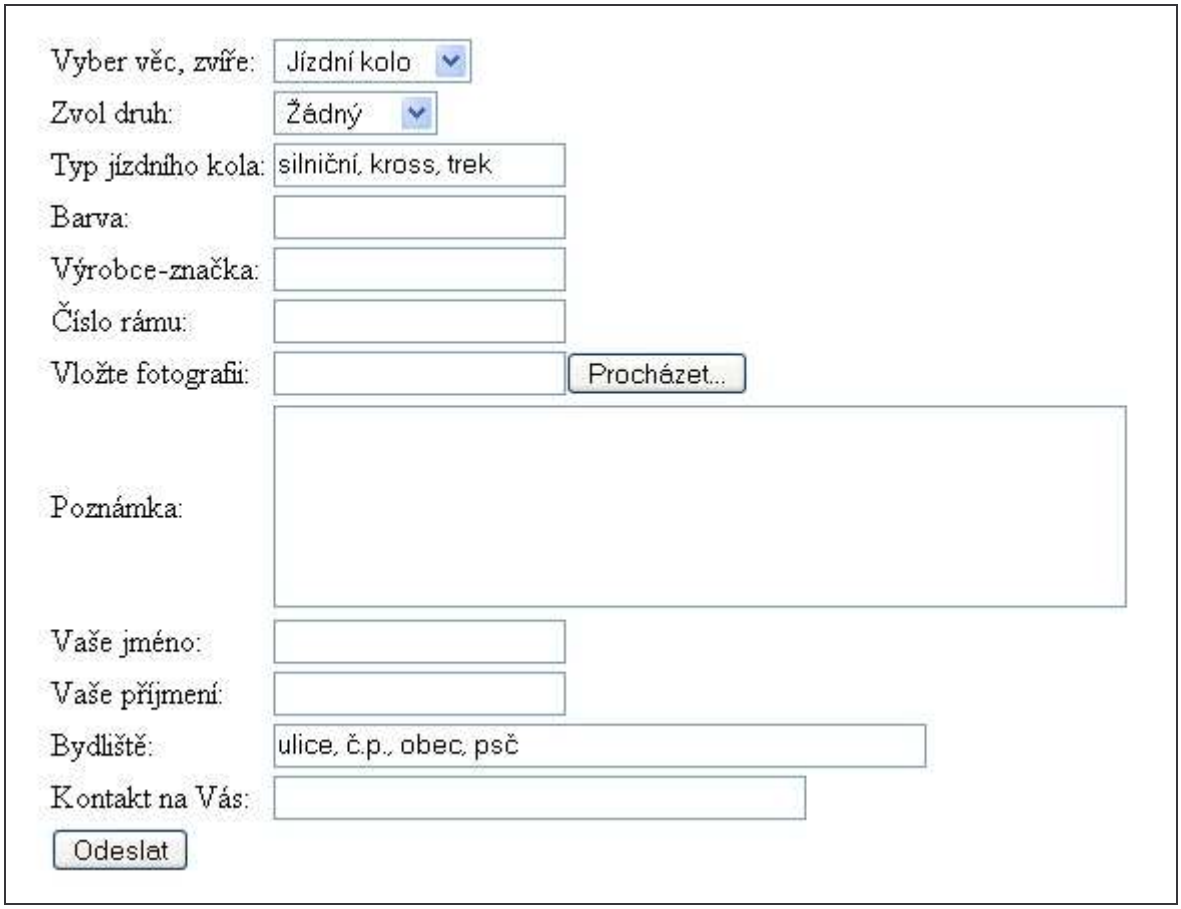
Doposud jsem hovořil jen o tom, jak může uživatel záznamy o ztrátách a nálezech hledat v databázi. Nezmínil jsem se ovšem, jak může onu ztrátu nahlásit. Tato možnost bude uživatelům poskytnuta na stránce: „Ztráty“ webové prezentace Městské policie Kyjov v sekci: „Ztráty nálezy“. Zde bude uživateli k dispozici formulář, pomocí kterého může zadat veškerá data o ztracené věci, nebo zvířeti. Kód formuláře je:

```
<FORM NAME="ztraty" METHOD="post" ACTION="hlaseni_ztrat.php"
ENCTYPE="multipart / form-data" >
Vyber věc, zvíře:
<SELECT NAME="vec">
        <OPTION VALUE="k">Jízdní kolo
        <OPTION VALUE="p">Peněženka
        <OPTION VALUE="kl">Klíče
        <OPTION VALUE="zv">Zvíře
        <OPTION VALUE="os">Jiné
</SELECT>
Zvol druh (jízdní kolo, peněženka)
<SELECT NAME="druh">
        <OPTION VALUE="">Žádný
        <OPTION VALUE="dámské">Dámské
        <OPTION VALUE="pánské">Pánské
        <OPTION VALUE="dětské">Dětské
</SELECT>
Typ jízdního kola:<INPUT TYPE=“text“ NAME="typ" VALUE=“ silniční, kross, trek“>
Barva:<INPUT  TYPE=“text“ NAME="barva">
Výrobce-značka:<INPUT TYPE=“text“ NAME="vyrobce">
Číslo rámu:<INPUT TYPE=“text“ NAME="cislo">
Vložte fotografii:<INPUT TYPE=“file“ NAME="foto">
Poznámka:< TEXTAREA rows="5" cols="50" NAME="zprava"></TEXTAREA>
Vaše jméno:<INPUT TYPE=“text“ NAME="jmeno">
Vaše příjmení:<INPUT TYPE=“text“ NAME="prijmeni">
Bydliště:<INPUT  TYPE=“text“  SIZE="50"  NAME="bydliste"  VALUE=“  ulice,  č.p.,
obec, psč“>
Kontakt na Vás:<INPUT TYPE=“text“ SIZE ="40" NAME="kontakt">
<INPUT TYPE=“submit“ VALUE="Odeslat">
</FORM>
```

Data z tohoto formulář bude zpracovávat skript: „*hlaseni_ztrat.php*“. Jeho kód je:

```
<?php
if ( Mail("nekdo@nekam.cz", $vec, $druh, $typ, $barva, $vyrobce, $cislo, $foto, $zprava,
"From: " . $jmeno, $prijmeni, $bydliste, $kontakt,) )
echo "Data byla odeslána";
```

else echo "Data se nepodařilo odeslat, zkuste to prosím znovu. V případě dalšího neúspěchu kontaktujte správce webových stránek."; ?>



Obrázek 5: formulář pro hlášení ztrát

Skript pro zpracování vložených dat do formuláře nedělá nic jiného, než že odešle vložená data na zadaný email. Tento způsob jsem zvolil hlavně z důvodu bezpečnosti. Povolit přímé vkládání dat z formuláře do databáze by bylo dost nebezpečné. Zvláště pokud chci, aby možnost nahlášení ztráty měl každý uživatel. Tedy i ten s nezabezpečeným přístupem. Z dosavadní praxe vyplívá, že hlášení ztrát zatím není využíváno v takové míře, aby je nestihl vkládat do databáze jeden zaměstnanec. Navíc je tu stále možnost nahlášení ztráty osobně, nebo telefonicky.

4.5 Přenesení na server

S administrátorem serveru, kde je umístěna webová prezentace Městské policie Kyjov, jsem se dohodl na způsobu přenesení mojí práce na server. Jelikož jedině administrátor

může na serveru vytvořit databázi a její uživatelské účty, vložím potřebné příkazy nadefinované v mojí diplomové práci do obyčejného textového souboru, a ten odešlu administrátorovi. Ten soubor s příkazy zkontroluje a zadá databázi MySQL příkaz pro provedení operací uložených v souboru. Poté se již mohu sám k databázi připojit prostřednictvím nástroje PHPmyAdmin. Tento nástroj se používá k administraci v MySQL. Jde o webové rozhraní k MySQL napsané v PHP. Pro připojení k databázi budu používat uživatelský účet označený jako: „spravce“. Budu mít tedy oprávnění k vytvoření a smazání tabulek databáze, vkládání a odstraňování dat, aktualizace dat a v neposlední řadě také k vykonání příkazů pro MySQL uložených v souboru. Právě tímto způsobem vytvořím v databázi jednotlivé její tabulky. Příkazy pro vytvoření tabulek si připravím do textového souboru jako tomu bylo při vytvoření souboru s příkazy pro administrátora serveru. Samotná data o nálezech a ztrátách vložím postupně do připravených tabulek databáze, opět pomocí nástroje PHPmyAdmin. Musím si ale dát pozor, aby PHPmyAdmin komunikoval s databází zvoleným kódováním znaků jako tomu bylo při vytvoření samotné databáze a poté také ve skriptech PHP. Data musí být také ve stejném kódování. Dosavadní evidence těchto dat bohužel není vedena elektronicky, proto provedu vložení dat ručně. Jednotlivé skripty PHP zakomponuji do stránek s HTML kódem tak, aby se vzájemně doplňovali a funkčně splňovali požadavky na ně kladené. Půjde o sadu vzájemně provázaných stránek, které budou zajišťovat úkoly zadané uživateli. Takto vytvořené stránky přenesu na webový server, pomocí protokolu FTP.

4.6 Bezpečnost

Bezpečnost je v dnešní době moderních technologií skloňována snad ve všech pádech. Není divu. Docela často se setkáváme s informací, že se někdo naboural do zcela bezpečného systému, že došlo k úniku osobních údajů a podobně. Nezřídka se jedná právě o důležité subjekty, jako jsou banky, státní instituce, výzkumné ústavy, tajné služby a nebo nemocnice. Bezpečnost je určitý kompromis mezi možností něco napsat, něco používat a zároveň aby to šlo spravovat. Máme tu čtyři strany, které mají každá jinou prioritu. Programátor by si přál, aby program uměl co nejvíce a aby šel napsat snadno a rychle. Uživateli se zamlouvá jednoduchá aplikace. To se samozřejmě vylučuje s bezpečností, které si uživatel příliš nevšímá do doby, než přijde o nějaká data, hůře peníze. Administrátor se nestará jen o jednotlivé aplikace, ale jeho úkolem je správa celého serveru. Snaží se zabezpečení jednotlivých aplikací nějak automatizovat, aby mu to nezabralo příliš mnoho času.

Poslední stranou je útočník. Typický útok není pravděpodobně veden za účelem likvidace systému, ale spíše nalezením jeho slabiny. Prolomením ochrany systému si útočník získá jméno. Útočník jako jediný ze všech čtyř stran, má na své straně čas.

Bezpečnost databáze je řešena originálním mechanismem přidělování práv. Lze nastavit kdo bude mít přístup k databázi a jaká oprávnění či omezení bude mít přidělena. Pokud chceme do databáze ukládat citlivá data, je možné data před vložením zašifrovat. Ovšem vždy je dobré pamatovat na skutečnost, že naroste čas při práci s databází a také naroste její objem. Tato metoda je vhodná, pokud do databáze ukládáme hesla uživatelů třeba nějakého portálu, kde je nutná registrace.Pro zašifrování můžeme použít funkci MD5.

Pokud vytvoříme aplikaci, která pracuje s databází, je bezpodmínečně nutné kontrolovat vstupní údaje vložené uživateli. Pokud tak neuděláme, vystavujeme se nebezpečí, že SQL dotazy, které naše aplikace generuje, budou mít jiný význam, než jsme zamýšleli. Neošetřené vstupní parametry může útočník zneužít ke vkládání fragmentů SQL dotazů. V některých případech (záleží na typu aplikace, použitém jazyku, databázovém engine) může útočník spouštět libovolné SQL příkazy, přepnout se i do jiné databáze a tam například smazat tabulky. Metodě, kdy někdo podvrhne falešné nebo nebezpečné parametry aplikaci, říkáme: „script injection“. Řešením je neustále kontrolovat vstupná data. V případě, že vstupní parametr známe, zkontrolujeme ho pomocí regulárních výrazů. V případě číselných vstupních hodnot je úprava zcela triviální. Vynásobíme číslo jedničkou, přičemž je možné i přičíst nulu. Lze ale předpokládat, že násobení bude rychlejší. Pokud je zadán nějaký řetězec, který začíná číslem, zbude jen hodnota počátečního čísla. Pokud je zadán řetězec nezačínající číslem, je výsledkem 0 a v tu chvíli bychom měli třeba přesměrovat na výchozí stránku a případně si nechat poslat na email, cože nám kdo do skriptu pustil. V případě textových vstupních parametrů je nutné patřičným způsobem uvést lomítkem (případně zdvojit) znak ' (apostrof) ve všech proměnných, které jsou vkládány do SQL řetězce. V PHP mohou být takto automaticky upraveny všechny řetězce z parametrů GET, POST a Cookies. PHP provádí potřebné vkládání lomítka při výchozím nastavení v souboru php.ini samo. Lze nastavit i automatické zdvojení apostrofu, nebo naopak jakékoli úpravy vypnout. Nechceme-li použít automatické ošetření nebo nemáme-li možnost jej aktivovat, pomůže nám funkce AddSlashes(), která přidává zpětná lomítka k zadanému řetězci.

PHP skripty velmi často také obsahují informace, které lze zneužít. Jedná se většinou o identifikační údaje pro přihlášení do systému, databáze, atd.. Tyto kódy sami o sobě žádný

problém nepředstavují, ale pokud by si je přečetl někdo neoprávněný, budeme mít velký problém. Tyto údaje se ve skriptech vyskytují většinou z důvodu porovnávání skutečných parametrů s parametry, které vloží uživatel. Přitom není nutné porovnávat samotná hesla. Můžeme porovnat jejich hash neboli otisk. Jedná se o šifru již zmíněné funkce MD5( ), která je jednosměrná. To znamená, že neexistuje způsob, jak z otisku vyluštit původní heslo.

Dalším způsobem, jak se může útočník dostat k našim identifikačním údajům je odposlouchávání komunikace mezi webovým prohlížečem a serverem. Útočníkovi stačí zachytit přenášená data, která právě vložil oprávněný uživatel při přihlašování do systému. Tato data se přenáší jako normální text a pro útočníka není problém tato data zneužít. Tento problém jde vyřešit použitím SSL. SSL (Secure Socket Layer) je nekomerční otevřený protokol, použitelný pro soukromé i komerční účely. Tímto protokolem lze zajistit šifrování přenášených dat a autentizaci serveru pomocí digitálních certifikátů. Pro použití SSL je třeba mít na straně serveru nainstalovanou podporu SSL a také jej musí podporovat webový prohlížeč, což v současnosti podporují téměř všechny. SSL přenášená data šifruje, čímž také narůstá jejich objem, prodlužují se časy pro přenos a navíc narůstá zatížení obou komunikujících stanic. Není tedy vhodné šifrovat veškerá přenášená data, ale jen ta, která jsou opravdu nejzranitelnější. [6]

Vytvořit stoprocentně bezpečný systém je prakticky nemožné. Vždy jde o balancování mezi riskováním a použitelností systému. Nejlepší bezpečnost je často nenápadná, dostatečně přizpůsobená požadavkům, bez toho aby bránila uživatelům dokončit jejich práci a bez toho aby autor přetěžoval kód extrémní složitostí. Nikdy však nezapomínejme na to, že: „Systém je pouze tak dobrý, jako je dobrý nejslabší článek v jeho řetězci“.

ZÁVĚR

Téma diplomové práce: „Přístup uživatelů do databáze MP Kyjov přes webové rozhraní“, plynule navazuje na moji bakalářskou práci, která se také týkala činnosti Městské policie Kyjov. V bakalářské práci jsem řešil problém, jak vhodným a hlavně výstižným způsobem prezentovat Městkou policii Kyjov a její činnost. Zvolil jsem proto způsob webové prezentace. Důraz jsem kladl na možnost vzájemné komunikace mezi občany a Městskou policií Kyjov. V diplomové práci jsem se tohoto základního pravidla snažil držet také. Bylo vyhodnoceno, že jako velice užitečná bude prezentace nalezených věcí na internetu. Později bylo rozhodnuto také o zařazení nahlášených ztrát právě spolu s nálezy. Místem zveřejnění bude již zmíněná webová prezentace Městské policie Kyjov. Na vyhodnocení vybraného záměru se podílelo několik stran. Své názory mohli vyjádřit samotní občané na webových stránkách, dále zaměstnanci Městské policie Kyjov a také představitelé města Kyjov.

Data o nálezech a ztrátách jsou uložena na serveru v databázi MySQL. Přistupovat k datům lze prostřednictvím webové prezentace Městské policie Kyjov. Prostředníkem a motorem mezi internetovým prohlížečem a serverem, kde je databáze uložena, je PHP. Uživatelé si mohou právě díky skriptům prohlížet data uložená v databázi. To k jakým datům mají uživatelé přístup je ošetřeno pomocí přidělených práv, které jsou pro jednotlivé skupiny uživatelů různá. Uživatelské účty jsou celkem tři. První má práva téměř zcela omezena. Smí prohlížet data a to jen z vybraných tabulek databáze. Data v těchto tabulkách jsou určena právě pro širokou veřejnost. Tento uživatelský účet není zabezpečen přihlašovacím heslem. Druhý uživatelský účet je již zabezpečen heslem. Důvodem je možnost prohlížení všech dat uložených v databázi. Toto oprávnění je určeno pro zaměstnance Městské policie Kyjov, kteří tak mohou nahlížet na soukromá data. Třetí uživatelský účet je k dispozici jen určenému člověku, který bude databázi aktualizovat. Bude tedy data odstraňovat, vkládat, upravovat, atd.. . Náplní diplomové práce bylo vytvořit funkční databázi MySQL a PHP skripty potřebné pro vykonání příkazů od jednotlivých uživatelů. Musel jsem vytvořit stránky s HTML kódem opatřené příslušnými prvky pro možnost vyhledávání potřebných dat dle zadané volby.

Již při výběru záměru bylo zahájeno jednání se zástupci okolních městských a obecních policí našeho regionu. Jednalo se hlavně o Městskou policii Hodonín, Veselí nad Moravou, Břeclav, Vracov a Obecní policii Ratíškovice. Spolupráce se státní policií ČR je samozřejmostí. Zmíněné organizace by vytvořily totožný systém a vedly svoji vlastní evidenci

nálezů a ztrát. Všechny tyto složky by měli druhý zmiňovaný uživatelský přístup a navzájem by tak mohli nahlížet na data uložená ve všech databázích. Tím by vzniklo větší územní pokrytí této služby pro občany, kteří by tak nemuseli kontaktovat jednotlivé složky zvlášť. Pokud by se systém v praxi osvědčil, mohla by se další jednání složek ubírat směrem k vytvoření centrálního registru a tím ke sjednocení všech databází do jedné. Občané by pak jistě uvítali možnost vyhledání jejich dotazu z jednoho místa na internetu. Bohužel zřízení takového centrálního registru není nijak jednoduchá záležitost, hlavně z hlediska ekonomického.

Dalším možným záměrem pro rozšíření služeb občanům ze strany vyjmenovaných složek by bylo vytvoření databáze registru jízdních kol. V případě že by registr byl centrální, byla by to neuvěřitelná zbraň proti zlodějům jízdních kol. Tato problematika se stává čím dál víc aktuálnější rok od roku. Napomáhá tomu i cena jízdních kol, kdy už není problém zakoupit jízdní kolo v řádu desetitisíců. Pokud by registr vznikl, a třeba ani ne centrální, byla by to bezpochyby alespoň výborná prevence.

ZÁVĚR V ANGLIČTINĚ

Conclusion:

The theme of my diploma work „ Access for users into the database of MP Kyjov through web interface „ , continously proceeds my bachelory work which was also related to activities of Kyjov Town Police.In my bachelory work I was resolving the problem of how to present Kyjov Town Police and its activities acceptable and mainly apposite way. In my diploma work I was trying to keep this basic rule, as well.There was evaluated, that very usefull would be a presentation of lost properties on internet. Later was decided to submit announced losses together with findings.The place of announcement would be the above mentioned web presentation of Kyjov Town Police.Several parties participated at evaluation of chosen intention.On web pages opinion of citizens of town Kyjov could have been expressed, further there have been used ideas of employees of Kyjov Town Police and Kyjov representatives.

Datas of lost properties are saved on a server in the database MySQL.The access to the datas is possible through the web presentation of Kyjov Town Police.The mediator and motor between an internet viewer and a server, where is the database saved is PHP.Users can browse datas saved in the database thanks to the scripts.The access to the datas is treated with a help of allocated rights, which are different for each group of users. There are three user´s accounts.The first one´s rights are nearly completely limited.It can inspect datas only from selected charts of database. Datas in these charts are reserved for general public.This user´s account is not secured by entry code.The second user´s account is already secured by entry code.The reason is a posibility to inspect all datas saved in the database.This licence is determined for employees of Kyjov Town Police, who can look through their personal datas.The third user´s account is available only for a specified person, who will update the database.He will delate, load, and edit the datas.The contens of my diploma work was to create functional database MySQL and PHP scripts needed for carring out orders from each user. There must have been created pages with HTML code secured with appropriate components for possibility of searching needed datas according to a required choice. During the selection of the intension dealing with representatives of surrounding Town police departments in our region began, especially with Town Police in Hodonín, Veselí nad Moravou, Břeclav, Vracov and Police department in Ratíškovice.The cooperation with Police of the Czech Republic is obvious.Above mentioned departments

would create identical system and would conduct their own evidence of lost properties. All of these departments would have the second, above mentioned, user´s access and together could inspect datas saved in all databases.Thereby there would be a bigger territorial coverage of this service for citizens, who would not have to contact each department extra.

If the system was proved good in working experience, dealing of departments could procced to creation of a central registry to unify of all databeses to one.Citizens would surely appreciate a possibility of searching their inquiries at one place on internet.Unfortunately, establishment of such a central registry is not an easy matter, mainly in economical term.

The other possible intention for extending services for citizens regarding named factors would be creating a bicycle registry. In case the registry was central, it would became an incredible weapon against thieves.This problem is becoming more and more actual.The price of bicycles helps to extend the problem, because it is not unusual to buy bicycle in ten thousands. If the registry was established, not necessarily the cenral one, it would be undoubtedly at least excelent prevention.

SEZNAM POUŽITÉ LITERATURY

[1] WELLING, L. THOMSONOVÁ, L. PHP a MySQL - rozvoj webových aplikací. Soft Press, 2004, ISBN: 80-86497-60-7.

[2] HOLČÍK, T. a kol. 1001 tipů a triků pro WWW stránky. Computer Press, 2006, ISBN: 80-7169-610-2.

[3] SCHNEIDER, R., D. MySQL - Oficiální průvodce tvorbou, správou a laděním databází. Grada, ISBN: 80-247-1516-3.

[4] ULLMAN, L. PHP a MySQL. Computer Press, Brno, 2004, ISBN: 80-251-063-4.

[5] KOSEK, J. PHP - tvorba interaktivních internetových aplikací. Grada Publishing, 1999, ISBN: 80-7169-373-1.

[6] Vývoj aplikací-databáze [online]. Dostupný z WWW: <http://interval.cz/vyvoj-aplikaci/databaze/> ISSN 1212-8651.

[7] MySQL-pestrý svět databází [online]. [cit. 2005-03-01]. Dostupný z WWW: <http://www.linuxsoft.cz/article.php?id_article=731>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

MySQL	Databáze komunikující pomocí jazyka SQL-Structured Query Language.
PHP	Personal Home Page, Hypertext Preprocesor.
HTML	Hyper Text Markup Language.
MD5	Message Digest algorithm 5.
FTP	File Transfer Protocol.
SSL	Secure Socket Layer.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK