Bakalářská práce

WEBová databáze pro publikace ve formátu BibTeX

Pavel Mejstřík 2006

Abstract

Bachelor work „Databaze for publication management in BibTeX format“ describes Internet application developement using MySQL databaze. This application is programed in PHP. This document covers business analysis of the application, its database design and final changes according to deteiled specification and at last PHP application program itself.

Abstakt

Bakalářská práce WEBová databáze pro publikace ve formátu BibTeX popisuje postup tvorby Internetové aplikace pracující s databází MySQL, která je naprogramována pomocí programovacího jazyka PHP, obsahuje rozbor zadání práce, návrh datové struktury, její upravení na základě zpřesněných požadavků a realizaci aplikace v programovacím jazyce PHP.

OBSAH

1. ÚVOD

Bakalářská práce Databáze pro správu publikací ve formátu BibTeX popisuje postup tvorby Internetové aplikace pracující s databází MySQL, která je naprogramována pomocí programovacího jazyka PHP, obsahuje rozbor zadání práce, návrh datové struktury, její upravení na základě zpřesněných požadavků a realizaci aplikace v programovacím jazyce PHP.

Téma své práce jsem si nevybral zcela náhodně. Již delší dobu se zajímám o Internet a možnost prezentace na něm. Přestože Internetové aplikace se v poslední době stávají samozřejmostí a většina prezentací se bez nich neobejde, je dle mého názoru málo programátorů, kteří jsou schopni psát kvalitní a rozsáhlé aplikace. V budoucnu bych se rád zabýval tvorbou prezentací na profesionální úrovni a proto jsem uvítal možnost věnovat se ve své bakalářské práci právě tvorbě internetové aplikace.

I výběr programovacího jazyka, kterým je PHP a databázového serveru MySQL, nebyl náhodný. Dle statistik uváděných na www stránkách je jejich používání velmi rozšířené a proto je jejich podpora na straně serveru téměř samozřejmostí.

Také zadané téma je vhodné pro získávání praktických zkušeností při tvorbě internetové aplikace. Mnoho z těchto aplikací se zabývá právě ukládáním a tříděním dat pomocí databázového serveru a jejich prezentací pomocí internetového prohlížeče.

Všechny tyto důvody ovlivnily výběr mé bakalářské práce a doufám, že poznatky, které jsem při jejím zpracování získal, budou přínosem pro můj pozdější profesní růst.

Na závěr bych rád poděkoval všem, kteří mi při zpracování mé bakalářské práce pomáhali a přispěli tak k jejímu kvalitnímu a odbornému zpracování.

2. ROZBOR ZADÁNÍ

2.1. První bod zadání

Úkolem prvního bodu zadání je provést obecný rozbor problému spojený s návrhem datového modelu vhodného pro uchování záznamů. Problém spolu s datovým návrhem má být diskutován na obecné úrovni, kde nebudeme uvažovat omezení daná zvolenými prostředky užitými k jejich dosažení. Jedná se pouze o teoretický rozbor problému sloužící k celkové analýze a ucelení pohledu na problém. Je nutné provést rozbor návrhu a jeho zpracování do přehledné formy pomocí ER diagramu. Dále je třeba vytvořit SQL příkazy pro práci s daným datovým modelem a provést diskusi nad jejich vhodností a možností dalšího rozšíření.

2.2. Druhý bod zadání

Při návrhu konkrétního řešení je nutné přizpůsobit obecný rozbor potřebám, možnostem a omezením, která plynou z výběru prostředku sloužícího k jejich dosažení. Je třeba se pokusit daná omezení eliminovat a jednoduše přizpůsobit datový návrh tak, aby bylo možné dosáhnout plné funkčnosti uvažované aplikace při zachování všech původních výhod datového modelu. Je nutné provést opět zpracování datového návrhu do přehledné formy ER diagramu a tvorbu všech příslušných příkazů. Dále je nutné provést diskusi nad daným návrhem a vysvětlení vhodnosti volby zvoleného řešení. Je nutné provést rozbor problémů při volbě jiných typů řešení a možnost jejich odstranění.

2.3. Třetí bod zadání

Posledním krokem realizace je přímo tvorba programu a zápis kódu programu, který je určen pro práci s daným datovým návrhem. Zde je nutné dbát na možné pozdější rozšíření aplikace a kvalitu okomentování celého programového kódu, které je nutné, aby bylo možné plně využít všech schopností a možností programu. Při psaní programu je nutné dbát na to, aby bylo vše na dostatečné bezpečnostní úrovni a přitom jasné a jednoduché. Ovládání aplikace musí být navrženo intuitivně, aby mohla být užívána bez nutnosti studovat její strukturu a programový kód.

3. REALIZACE PRVNÍHO BODU ZADÁNÍ

3.1. Základní seznámení s BibTeXem

BibTeX je program a formát souboru navržený Oren Patashnikem a Leslie Lamportem v 1985 pro ukládání dat o LaTeX dokumentech. Formát je plně znakově založený a může být užívaný v jakémkoliv programu. Formát pracující s poli a programem zpracovávajícím BibTeXový soubor bude neznámá pole ignorovat. Jedná se o nejběžnější formát pro bibliografii na Internet.

Ve standardu formátu BibTeX je definováno několik základních typů a polí (field). Každý dokument lze zařadit do jednoho z těchto typů. Každý z těchto typů má předepsaná pole (field), která musí obsahovat a jsou povinná. Dále jsou standardem definována i pole doporučená, která by měl daný typ obsahovat, ale jejich nepřítomnost je povolena. Přehled jednotlivých typů polí a jejich vztahů je v následujících tabulkách.

Tabulka 1: Seznam standardních polí

Jméno pole	Stručný popis pole
Address	Obvykle adresa vydavatele, nakladatelství nebo jiné instituce. Větším nakladatelstvím doporučuje Van Leunen tuto informaci vynechat. Na druhé straně malí vydavatelé mohou udáním své adresy čtenáři pomoci.
Annote	Poznámka. Nepoužívá se u standardních bibliografických stylů, ale může být užita v ostatní tvorbě okomentovaných bibliografií.
Author	Jméno(a) autora(ů), ve formátu popsaném v LaTeXové knize.
Booktitle	Titul knihy, jejíž část je citována.
Chapter	Číslo kapitoly, části nebo čehokoliv jiného.
Crossref	Databázový klíč vstupu odkazující na jinou stránku. Některá pole, která chybějí ve zpracovávaném záznamu jsou odvozena od pole, které je z odkazu na jinou stránku.
Edition	Pořadí vydání knihy, například "Second" (druhé). Tato číslovka by měla patřit mezi řadové a měla by být zapsána s prvním písmenem velkým.
Editor	Jméno(a) vydavatele(ů), typ označený LaTeXové knize. Jestliže je také pole Author, pak pole Editor obsahuje vydavatele knihy nebo kolekce.
Howpublished	Jak bylo něco jiného vydáno. První slovo by mělo být napsáno velkým písmem.
Institution	Garant technické zprávy.
Jurnal	Jméno časopisu. Zkrácení jsou poskytnuta pro mnoho deníků.
Key	Používá se k abecednímu řazení, odkazu na jinou stránku a k vytvoření štítku, pokud je autor neznámý. Toto pole by se nemělo zaměnit s polem klíčů užívaných v "cite" příkazu a na začátku databázového vstupu.
Month	Měsíc vydání nebo u ještě nezveřejněné práce měsíc, kdy byla napsána. Lze použít standardu třípísmenného zkrácení podle popisu v Appendix B.1.3 LaTeXové knihy.
Note	Jakákoliv dodatečná informace, která by mohla čtenáři pomoci. První slovo by mělo být psáno velkým písmem.
Number	Číslo deníku, časopisu, technické zprávy nebo práce z nějaké série. Vydání deníku nebo magazínu je obvykle udáno jejím dílem a číslem; organizace vydávající technické zprávy je obvykle číslují a v některých případech jsou knihám dána čísla udávající jejich pořadí v sérii.
Organization	Organizace, která sponzoruje danou konferenci nebo publikuje manuál.
Pages	Jedno nebo více čísel stránek nebo rozsah čísel, jako 42, 111 nebo 7, 41, 73-97 či 43+ ('+' v posledním příkladě znamená „všechny strany následující po 43 “). Pro zlepšení udržení kompatibility s databází se ve standardních stylech mění jednoduchá pomlčka (7-33) v pomlčku dvojitou (7--33) používanou v TeX.
Publisher	Jméno vydavatele.
School	Jméno školy, kde byla teze napsána.
Series	Název řady nebo sady knih. Když je citována celá kniha, pole „title“ udává titul knihy a nepovinné pole „series“ může udávat jméno série nebo vícesvazkové řady, ve které byla kniha vydána.
Title	Titul práce, typ je vysvětlen v LaTeXové knize.
Type	Typ technické zprávy, například "Research Note".
Volume	Název svazku deníku nebo vícesvazkové knihy.
Year	Rok vydání nebo u ještě nezveřejněné práce rok, kdy byla napsána. Může se skládat ze čtyř číslovek, jako 1984, ačkoli standardní styly mohou pracovat s jakýmkoliv rokem, jehož poslední čtyři neinterpunkční znaky jsou číslovky, například '\hbox{(about 1984)}'.

Tabulka 2: Seznam ostatních polí

Jméno pole	Stručný popis pole
Affiliation	Spoluautoři.
Abstract	Obsah práce.
Contents	Obsah.
Copyright	Autorská práva.
ISBN	The International Standard Book Number.
ISSN	The International Standard Serial Number. Užívaný k identifikaci časopisu.
Keywords	Klíčová slova se používají převážně pro vyhledávání nebo eventuelně i pro poznámku.
Language	Jazyk, ve kterém je dokument napsán.
Location	Místo, kde daný vstup vznikl - obvykle místo, kde se konala daná konference.
LCCN	The Library of Congress Call Number. Také se může použít „lib-congress“.
Mrnumber	The Mathematical Reviews number.
Price	Cena dokumentu.
Size	Rozsah dokumentu.
URL	WWW Universal Resource Locator, který určuje reference pro článek. Často se používá pro technické zprávy nebo dodatky na ftp adrese, kde je umístěn dodatek k dokumentu.

Tabulka 3: Standardní vstupní typy a jim odpovídající pole

Vstupní typ	Povinná pole Doporučená pole	Popis
@article	author, title, journal, year	Článek z časopisu nebo magazínu.
	volume, number, pages, month, note, key	
@book	author or editor, title, publisher, year	Kniha s jasným vydavatelem.
	volume, series, address, edition, month, note, key	
@booklet	Title	Práce je tištěná a vázaná , ale bez jmenovaného vydavatele nebo garanta.
	author, howpublished, address, month, year, note, key	
@collection		Sbírka prací. Stejný jako @proceedings.
@conference	author, title, booktitle, year	Stejný jako @inproceedings.
	editor, pages, organization, publisher, address, month, note, key	
@inbook	author or editor, title, chapter and/or pages, publisher, year	Část knihy, která muže být kapitola (nebo úsek) a/nebo rozmezí stránek.
	volume, series, address, edition, month, note, key	
@incollection	author, title, booktitle, year	Součást knihy, mající svůj vlastní název.
	editor, pages, organization, publisher, address, month, note, key	
@inproceedings	author, title, booktitle, year	Článek zápisu z jednání konference.
	editor, pages, organization, publisher, address, month, note, key	
@manual	title	Technická dokumentace.
	author, organization, address, edition, month, year, note, key	
@mastersthesis	author, title, school, year	Master's zásada.
	address, month, note, key	
@misc		Užívá se tento typ, pokud není nic vhodnějšího.
	author, title, howpublished, month, year, note, key	
@patent		Patent.
@phdthesis	author, title, school, year	PhDr. teze.
	address, month, note, key	
@proceedings	title, year	Protokoly porady.
	editor, publisher, organization, address, month, note, key	
@techreport	author, title, institution, year	Referát vydaný školou nebo jiným ústavem, obyčejně číslován v řadě.
	type, number, address, month, note, key	
@unpublished	author, title, note	Dokument mající autora a titul, ale není formálně vydaný.
	month, year, key	

3.2. Implementace standardu BibTeX

Z uvedených tabulek je zřejmé, že jakýkoliv dokument lze přiřadit do jednoho z typů uvedených standardem. Každý z typů má přidělena určitá pole, která jsou považována za povinná a doporučená a musí být vyplněna, aby byly splněny podmínky standardu. Databáze proto musí být postavena tak, aby bylo možné zajistit kontrolu těchto pravidel daných standardem. Proto je nutné vést údaje nejen o jednotlivých dokumentech, ale i o typech, polích a jejich vztazích. Pravidlem také je, že je-li pole u daného typu povinné, nemůže být zároveň doporučené. Z toho plyne, že dané pole smí mít pro určitý typ pouze jedno doporučení.

Ze zadání práce je zřejmé, že každý záznam v jednom poli má pevně danou maximální délku a není nijak obsahově omezen. U každého dokumentu je také nutné vést v evidenci "referenci", což je jedinečný záznam určený pro identifikaci dokumentu v knihovně. U dokumentu není nijak omezeno jaká z polí budou vyplněna, kromě povinnosti vyplnění povinných polí. Množství jednotlivých polí uložených u daného dokumentu tedy není pevně stanoveno a je proměnné.

3.2.1. Základní rozdělení

Databáze bude rozdělena na dvě hlavní části. První z nich bude uchovávat informace o typech, polích a jejich vztazích. Druhá část bude určena pro uchovávání dat o jednotlivých dokumentech.

3.2.2. První část

Pro potřeby uchovávání informací o typech je nutné, aby byla v databázi tabulka, ve které bude uveden úplný seznam všech povolených typů. Každý ze záznamů bude mít své identifikační číslo, které bude primárním klíčem tabulky. Další z tabulek bude obsahovat úplný seznam všech polí spolu s identifikačním číslem, které bude opět primárním klíčem.

TABLE field	
id_field	**field**
1	Author
2	Editor
⋮	⋮

TABLE type	
id_type	**type**
1	@article
2	@book
⋮	⋮

Obr 1: Schéma tabulek field a type

3.2.2.1. Uložení vztahů typů a polí

Pro definici vztahů mezi typy a poli je možné zvolit různé druhy řešení. Lze vytvořit jednu tabulku, kde bude identifikační číslo typu, identifikační číslo pole a hodnota určující v jakém vztahu jsou tyto položky. Výhodou tohoto řešení je, že se jedná o jedinou tabulku a veškerá kontrola záznamů probíhá právě jen v této tabulce. Požadavek na jedinečnost kombinace identifikátoru pole a typu bude kontrolována přímo integritním omezením databáze, čímž se snižuje množství nutných kontrol ze strany aplikace.

TABLE field		
id_field	**id_type**	**vztah**
1	1	req
2	2	opt
⋮	⋮	⋮

Obr 2: Schéma tabulky vztahů

TABLE type			
id_type	**type**	**req**	**opt**
1	@article	1,2,5	5,8
2	@book	2,7,3	11
⋮	⋮	⋮	⋮

Obr 3: Schéma tabulek vztahů 2

Dalším možným řešením je, že se seznam povinných a doporučených polí přidá do tabulky typu. Nevýhodou tohoto řešení je, že neznáme předem počet povinných,, resp. doporučených polí u daného typu a je tedy nutné zavést do tabulky všechny možné kombinace. Kontrola daného sloupce je také velmi náročná, protože se u každého typu liší počet doporučených a povinných polí. Provádět kontrolu v takovéto tabulce by bylo poměrně náročné s ohledem na tvorbu SQL dotazů.

Třetím možným řešením je zavést dvě tabulky, z nichž jedna bude sloužit pro správu povinných polí a druhá pro správu doporučených polí. V obou tabulkách budou vedena pouze identifikační čísla typu a pole, mezi nimiž daný vztah platí. Kontrola, zda není dané pole u jednoho typu uvedeno dvakrát, pak musí probíhat právě mezi těmito

TABLE t_f_req	
id_type	**id_field**
1	1
1	2
⋮	⋮

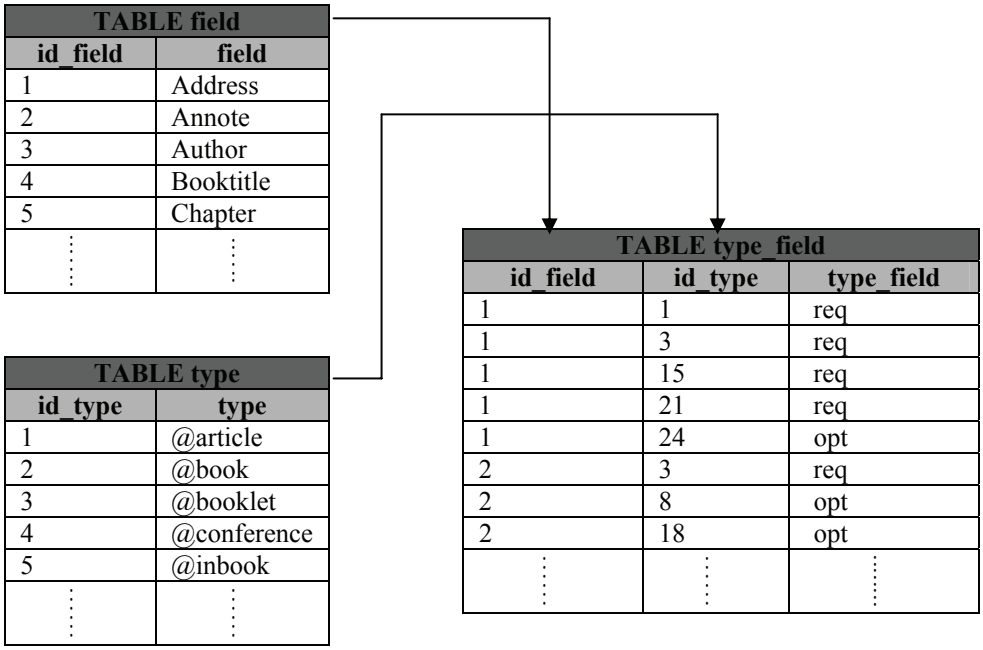
TABLE t_f_opt	
id_type	**id_field**
2	3
5	18
⋮	⋮

Obr 4: Schéma tabulek vztahů 3

dvěma tabulkami. Výhodou tohoto řešení je, že identifikační číslo pole a typu lze považovat za unikátní klíč a odpadá tak nutnost kontrolovat duplicitu v jedné tabulce, neboť je zajištěna samotnou databází.

Jistě by bylo možné navrhnout několik dalších řešení pro tento problém, ale dle mého názoru je nejvhodnějším řešením první navrhované. Jeho nespornou výhodou je jasná definice unikátního klíče a tím snadné zajištění integrity. Provádění kontroly, zda se jedno pole nevyskytuje u typu jako povinné i doporučené, je poměrně snadné a lze ho provádět jediným dotazem.

Na základě tohoto rozhodnutí lze již navrhnout první hlavní část databáze sloužící k uchovávání informací o typech, polích a jejich vztazích. Půjde tedy o čtyři tabulky. První tabulka uchovává informace o typech a bude obsahovat identifikační číslo a název typu. Druhá tabulka bude sloužit k uchování informací o polích a bude obsahovat identifikační číslo a název pole. Třetí tabulka bude uchovávat informaci o vztahu mezi polem a typem, ve kterém se nachází. Atributy budou identifikační číslo typu a identifikační číslo pole a vztah definovaný z omezené skupiny dané databází.



Obr 5: Návrh ER-diagramu pro první část databáze

3.2.3. Druhá část

Při řešení druhé hlavní části databáze sloužící k uchovávání údajů o jednotlivých dokumentech se objevuje také několik omezení. První je dané zadáním a stanovuje, že žádný ze záznamů nepřesáhne délku 1000 znaků. Další omezení vyplývá z povinnosti vyplnění povinných polí. Na toto pravidlo však není nutné dbát při návrhu datového modelu, ale až při návrhu samotné aplikace zpravující data tohoto návrhu. Při návrhu datového modelu je však nutné uvážit, zda obsah jednotlivých polí bude možné zadávat bez omezení nebo budou k dispozici předdefinované hodnoty, které se budou pouze přiřazovat. Toho by bylo možné dosáhnout dvěma způsoby. Buď předdefinováním hodnot přímo při tvorbě tabulky nebo použitím pomocné tabulky, na kterou by se dané pole záznamu odkazovalo. První možnost má pouze velmi omezené použití, protože je nutné všechny možnosti pro dané pole definovat již při samotném návrhu databázové struktury nebo později zasahovat do této struktury. Druhá možnost nám dovoluje tento problém obejít a lze další předdefinované hodnoty později přidávat. V případě, že nebudeme používat žádné předdefinované hodnoty, je nutné zvýšit důraz kladený na kontrolu vkládaných dat a jejich úplnost.

S ohledem na strukturu dat ukládaných v databázi je vhodné zvolit možnost bez předdefinovaných obsahů polí, protože není možné dopředu stanovit obsah těchto polí. Předdefinování hodnot by bylo možné použít jen u některých druhů polí, ale nebylo by možné definovat všechny možnosti a proto je výhodnější použití přímého vstupu dat bez jejich předchozího předdefinování.

Při návrhu datové struktury lze opět použít několik řešení. Prvním z nich je možnost vedení jedné tabulky, ve které budou atributy identifikační číslo daného dokumentu, reference vedená pro potřeby vyhledávání, identifikační číslo typu dokumentu a dále pro každé existující pole by existoval jeden

TABLE dokument				
id_dokument	**reference**	**Field_1**	**Field_2**	**……..……**
1	Knihaautor:85	Autor	kniha	……..……
2	Oracle:98	Angel	Oracle	……..……
3	Práce:99	Proler	Práce o knize	……..……
⋮	⋮	⋮	⋮	⋮

Obr 6: Schéma tabulky dokument

atribut. Toto řešení není výhodné, neboť tabulka by byla velmi rozsáhlá a docházelo by ke zbytečné alokaci prostoru pro pole, která by nebyla nikdy vyplněna. Také v případě přidání nového pole by bylo nutné zasahovat do struktury tabulky a přidávat nový sloupec. Výhodou tohoto řešení je velmi jednoduché vyhledávání v takovéto tabulce a daly by se tvořit poměrně náročné dotazy pro vyhledávání.

Další možností je vytvořit jednu tabulku určenou pro uchovávání základních údajů o dokumentu. V této tabulce by se nacházely atributy identifikační číslo dokumentu, reference a identifikační číslo typu dokumentu. Poté by existovala další tabulka, ve které by bylo uloženo identifikační číslo dokumentu, identifikační číslo pole a obsah pole pro daný dokument odpovídající danému poli. V takto zavedených tabulkách jsou uváděna pouze pole, která byla uživatelem definována a odpadá tak nutnost zbytečné alokace datového prostoru. Nevýhodou takovéhoto řešení je nutnost spojování tabulek při vyhledávání a tím i náročnější dotazy a vyšší nároky na databázový server.

Je také možné použít tabulku pro daný dokument tak, jak byla uvedena v předchozím příkladu, která by obsahovala identifikační číslo daného dokumentu, dále referenci a identifikační číslo typu dokumentu. Dále by existovalo několik tabulek tak, že každá by odpovídala existujícímu poli a v těchto tabulkách by bylo vedeno vždy identifikační číslo dokumentu a obsah pole pro daný dokument. Toto řešení má nevýhodu ve velkém množství tabulek a také v nutnosti tvorby nových tabulek v případě přidání nového pole. Pro vyhledávání opět platí, že by bylo nutné používat spojování tabulek a dotazy by tak byly velmi náročné.

TABLE dokument		
id_dokument	**reference**	**id_type**
1	Knihaautor:85	2
2	Oracle:98	2
3	Práce:99	5
⋮	⋮	⋮

TABLE data_dokumentu		
id_dokument	**id_field**	**obsah_field**
1	2	Angel
1	4	Oracle
3	4	Práce o knize
⋮	⋮	

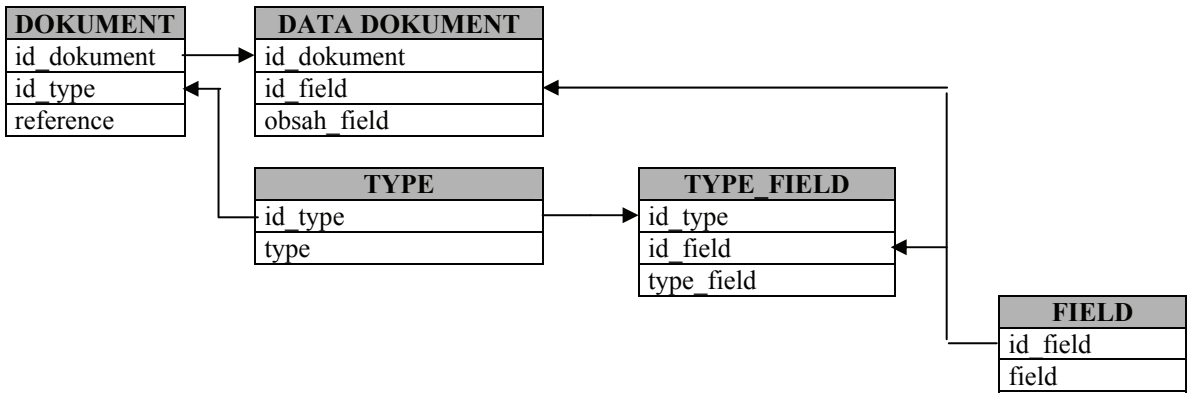
Obr 7: Schéma tabulky dokument a data dokumentu

Opět lze navrhnout ještě několik možných řešení a v tomto případě nelze stanovit, které z řešení je ideální. Při volbě řešení bude záležet na množství ukládaných dat, jejich struktuře a obsahu. Budou-li ukládaná data konzistentní a u většiny dokumentů budou vyplněna téměř všechna nebo dokonce všechna pole, bylo by vhodné volit první z navrhovaných řešení. Druhé a třetí řešení mají svou výhodu naopak v případě, že nepředpokládáme takovouto úplnost dat. Druhé řešení oproti zbývajícím má nespornou výhodu, že přidávání nových polí nemá vliv na datovou strukturu a není nutné ji měnit, ať již přidáváním a ubíráním sloupců v tabulce nebo přímo přidáváním a mazáním celých tabulek.

Vzhledem k této výhodě a struktuře dat jsem za vhodné řešení vybral druhé, tedy dvě tabulky, kde jedna z nich slouží k uchování základních údajů o dokumentu a druhá k uchovávání údajů o obsazích polí jednotlivých dokumentů.

3.2.4. Navrhovaná databázová struktura

Na základě všech předchozích rozhodnutí lze již navrhnout druhou část databázového návrhu. Půjde tedy o dvě tabulky, kde první bude sloužit k uchování základních údajů o dokumentu a druhá k uchovávání údajů o obsazích polí jednotlivých dokumentů. Tyto obsahy se budou vkládat přímo a nebudou nijak předdefinovány ani omezeny datovým návrhem.



Obr 8: Návrh ER-diagramu databáze

4. REALIZACE DRUHÉHO BODU ZADÁNÍ

4.1. Základní seznámení s MySQL

MySQL byl vyvinut společností MySQL AB. MySQL AB je obchodní společnost, zabývající se poskytováním služeb souvisejících s databází MySQL. Společnost byla založena ve Švédsku, Švédy Davidem Axmarkem, Allenem Larssonem a Finem Michaelem "Montym" Wideniusem, kteří spolu pracovali již od 80. let. MySQL AB je jediným vlastníkem MySQL server kódu, ochranné známky MySQL a mysql.com domény na světě. Společnost je v osobním vlastnictví a bez závazků.

4.1.1. Stávající verze

Aktuální verze MySQL v době návrhu byla verze 4.1.13. Při vývoji aplikace bylo nutné brát v potaz veškerá omezení a možnosti této verze a celá aplikace je tedy navrhována tak, aby všech výhod dané verze bylo využito.

4.1.2. Proč používat MySQL

Hlavním důvodem pro použití serveru MySQL je to, že jde o open source, který lze na Internetu získat zdarma. MySQL je silně podporován stejně jako PHP a dochází k jeho prudkému rozvoji na Internetu. Díky velkému zájmu a širokému využití je velice rychle vyvíjen a v brzké době se lze dočkat toho, že bude srovnatelný s jinými databázovými servery. MySQL je přímo určen pro práci na Internetu a na zpracovávání mnoha dotazů v jednom okamžiku a tak je jedním z nejlepších serverů, které lze pro účely Internetu použít. Ostatní servery mají mnoho funkcí, které však na Internetu nejsou nezbytné a zbytečně komplikují celý databázový server. Tato rozšíření jsou vhodná pro komplikované podnikové databáze, rozsáhlé knihovny apod., ale nejsou potřebná pro webovské aplikace. Právě proto, že MySQL neobsahuje tyto rozšířené funkce, lze jej získat zdarma oproti jiným databázovým serverům, jejichž cena je velice vysoká. Všechny tyto důvody jsou podstatné a předurčují MySQL pro používání na Internetu.

4.2. Analýza navrhované datové struktury

4.2.1. Tvorba datového prostoru

Jedná se o část dotazovacího jazyka, která se obecně nazývá DDL (Data Definition Language). Jde o příkazy sloužící k definování datových struktur,, resp. objektů.

Při návrhu souborů sloužících k tvorbě datového prostoru nebylo třeba datový návrh nijak upravovat a provádět v něm změny. Došlo však k diskutování datových typů pro jednotlivé atributy použité v tabulkách. Pro identifikační čísla, která slouží jako primární klíče byl použit typ integer. Pro jednotlivé publikace je délka identifikačního čísla šest znaků a pro ostatní pak čtyři znaky. Tato délka byla zvolena s ohledem na předpokládané množství dat. Publikací se tak může v databázi nacházet až milion, což je mnohem více než zadané množství publikací, které se budou v databázi nacházet. Volba tohoto množství však slouží pro případ pozdějšího rozšíření celé aplikace. U typů a polí je délka identifikačního čísla čtyři znaky také velmi předimenzována, ale množství dat, které je ukládáno nemůže mít vliv na velikost databáze ani na rychlost její odezvy na dotaz.

4.2.1.1. Tabulka DOKUMENT

V tabulce DOKUMENT je dle datového návrhu uloženo identifikační číslo publikace o délce šest znaků typu integer, dále identifikační číslo typu o délce čtyři znaky typu integer. Dalším atributem je reference. Pro tento atribut jsem zvolil typ varchar a délku padesát znaků. Tento atribut slouží k ukládání informace o základních informativních údajích o daném dokumentu. Délka padesát znaků pro tento atribut byla zvolena po konzultaci dle zkušeností na požadovanou délku ukládané informace. Pro omezení atributů platí, že identifikační číslo dokumentu nesmí mít hodnotu NULL a je zde použito funkce MySQL autoincremet, která zajišťuje automaticky navýšit poslední použité číslo o jedničku při vkládání nového řádku do tabulky. Pro atribut identifikační číslo typu platí, že nesmí být vyplněna hodnota NULL a jako defaultní nastavení byla zvolena hodnota nula. Atribut reference má také omezení zakazující vložení hodnoty NULL a nemá definovanou žádnou defaultní hodnotu. Jako primární klíč tabulky je použito identifikační číslo dokumentu.

DOKUMENT			
Atribut	**Typ atributu**	**Hodnota NULL**	**Výchozí hodnota**
id_dokument	integer (6)	NOT NULL	auto increment
id_type	integer (4)	NOT NULL	0
reference	varchar (50)	NOT NULL	

```
CREATE TABLE dokument (
      id int(6) NOT NULL auto_increment,
      id_type int(4) NOT NULL default '0',
      reference varchar(50) NOT NULL,
      PRIMARY KEY  (id)
);
```

Obr 9: Struktura tabulky DOKUMENT a SQL dotaz pro její vytvoření

4.2.1.2. Tabulka DATA DOKUMENT

V tabulce DATA DOKUMENT je uloženo identifikační číslo dokumentu, ke kterému daná data patří. Toto identifikační číslo má délku šest znaků typu integer tak jako identifikační číslo z tabulky DOKUMENT. V tabulce je dále identifikační číslo typu, ke kterému se data vztahují. Identifikační číslo má délku čtyři znaky typu integer. Dalším atributem je field_text, který slouží k uložení dat daného typu pro danou publikaci. Byl zvolen typ text sloužící v MySQL k ukládání dlouhých znakových dat. Typ varchar nelze použít, protože maximální povolená délka je 255, což neodpovídá zadaným požadavkům, které počítají s 1000 znaky. Žádný z atributů nemá povolenou hodnotu NULL. Pro identifikační čísla je nastavena defaultní hodnota na nulu a pro atribut field_text není nastavena žádná defaultní hodnota. Klíč je v této tabulce typu UNIQUE složený z identifikačního čísla dokumentu a identifikačního čísla typu.

DATA DOKUMENT			
Atribut	**Typ atributu**	**Hodnota NULL**	**Výchozí hodnota**
id_dokument	integer (6)	NOT NULL	0
id_type	integer (4)	NOT NULL	0
field_text	text	NOT NULL	

```
CREATE TABLE data dokument (
      id_dokument int(6) NOT NULL default '0',
      id_field int(4) NOT NULL default '0',
      field_text text NOT NULL,
      UNIQUE KEY id_dokument (id_dokument,id_field)
);
```

Obr 10: Struktura tabulky DATA DOKUMENT a SQL dotaz pro její vytvoření

4.2.1.3. Tabulka FIELD

V této tabulce jsou uloženy údaje o polích. Slouží k tomu atribut identifikační číslo pole, které má délku čtyři znaky a je typu integer. Druhým atributem tabulky je atribut field délky patnáct znaků typu varchar. V atributu je uložen název daného pole. Žádný z atributů nemá povolenou hodnotu NULL a pro atribut identifikační číslo je použita funkce MySQL autoincrement. Identifikační číslo pole slouží jako primární klíč.

FIELD			
Atribut	**Typ atributu**	**Hodnota NULL**	**Výchozí hodnota**
id_field	integer (4)	NOT NULL	auto increment
field	varchar (15)	NOT NULL	

```
CREATE TABLE field (
      id_field int(4) NOT NULL auto_increment,
      field varchar(15) NOT NULL default '',
      PRIMARY KEY  (id_field))
);
```

Obr 11: Struktura tabulky FIELD a SQL dotaz pro její vytvoření

4.2.1.4. Tabulka TYPE

Tabulka slouží k uchovávání dat o typech. Její struktura je stejná jako u tabulky FIELD. Prvním atributem je zde identifikační číslo typu, které má délku čtyři znaky a je typu integer. Druhý atribut type je typu varchar o délce patnáct znaků a slouží k uchování názvu daného typu. Ani jeden atribut nesmí nabývat hodnoty NULL a pro identifikační číslo je použita funkce MySQL autoincrement. Identifikační číslo typu je také primárním klíčem tabulky.

TYPE			
Atribut	**Typ atributu**	**Hodnota NULL**	**Výchozí hodnota**
id_type	integer (4)	NOT NULL	auto increment
type	varchar (15)	NOT NULL	

```
CREATE TABLE type (
      id_type int(4) NOT NULL auto_increment,
      type varchar(15) NOT NULL default '',
      PRIMARY KEY  (id_type)
);
```

Obr 12: Struktura tabulky TYPE a SQL dotaz pro její vytvoření

4.2.1.5. Tabulka TYPE_FIELD

Tabulka slouží k uložení vztahů mezi typy a poli. Prvním atributem tabulky je identifikační číslo mající délku čtyři znaky typu integer. Druhý atribut je identifikační číslo pole mající také délku čtyři znaky typu integer. Posledním atributem je atribut, který může nabývat hodnoty ‚req' neboli povinné pole (require), ‚opt' doporučené pole (optional) nebo ‚ost', což jsou ostatní. Všechny atributy mají zakázanou hodnotu NULL. Identifikační čísla jako defaultní hodnotu používají nulu a atribut používá jako defaultní hodnotu nastavení ‚ost'. Jako klíč je použit klíč typu unique složený z atributů identifikační číslo typu a identifikační číslo pole.

TYPE_FIELD			
Atribut	**Typ atributu**	**Hodnota NULL**	**Výchozí hodnota**
id_type	integer (4)	NOT NULL	0
id_field	integer (4)	NOT NULL	0
type_field	enum (req,opt,ost)	NOT NULL	ost

```
CREATE TABLE type_field (
      id_type int(4) NOT NULL default '0',
      id_field int(4) NOT NULL default '0',
      UNIQUE KEY id_type__id_field_uk (id_type,id_field)
);
```

Obr 13: Struktura tabulky TYPE_FIELD a SQL dotaz pro její vytvoření

4.2.2. Tvorba dotazů pro práci s daty

Tato část dotazovacího jazyka se nazývá DML (Data Manipulation Language), což je skupina příkazů, která slouží k manipulování s daty, tedy k jejich přidávání, upravování, mazání a prohledávání.

Dotazy pro práci s daty lze rozdělit do několika základních skupin. První možné dělení je na dotazy dle jejich funkce, tedy dotazy INSERT, UPDATE, DELETE a SELECT. Dalším možným rozdělením je dle jejich použití v budoucí aplikaci. Toto dělení považuji za výhodnější pro rozbor a analýzu a proto ho budu používat. Nejdříve je nutné zpracovat dotazy pro správu první části databáze dle návrhu. Zde se jedná o dotazy, které budou sloužit pro úpravu dat v tabulkách TYPE, FIELD a TYPE_FIELD. V druhé části databáze bude nutné zpracovat dotazy umožňující přidávání nových dat, úpravu dat již vložených a také vyhledávání v těchto datech. Data jsou uložena v tabulkách DOKUMENTY a DATA DOKUMENTY.

4.2.2.1. Dotazy pro první část databáze

Při správě dat o typech, polích a jejich vztazích bude nutné provádět jejich výpis, vkládání , úpravy a mazání. Na práci s daty je celá práce zaměřena a proto zde podrobně vyložím jak dotazy vypadají a vysvětlím jejich obsah. V místech, kde je v dotazu zapsána značka ??? je nutné vložit požadovaný údaj. Tento údaj ovlivňuje výsledek dotazu a bude zadáván programem dle potřeby.

K prohlížení dat slouží pouze dva dotazy SELECT. Tyto dotazy zobrazí všechny záznamy z tabulek FIELD a TYPE. V případě potřeby je možné omezit dotaz podmínkou WHERE, kde lze zadat identifikační číslo, které pro nás bude zajímavé. Dále lze také použít klauzuli ORDER BY sloužící k seřazení záznamů. Pro nás má v tomto případě význam řadit podle textové části field nebo type, která seřadí záznamy abecedně.

```
SELECT id_field, field FROM field WHERE id_field=??? ORDER BY field;
SELECT id_type, type FROM type WHERE id_type=??? ORDER BY type;
```

Dalším dotazem je nutné zobrazit všechna pole, která jsou povinná, doporučená nebo ostatní pro daný typ. Identifikační číslo hledaného typu se vloží do klauzule WHERE spolu s textem označujícím o jaké pole máme zájem. Získaná pole je opět možné řadit klauzulí ORDER BY.

```
SELECT field.id_field, field.field FROM type_field, field
      WHERE ((id_type=??? AND atribut=???)
            AND field.id_field=type_field.id_field)
      ORDER BY id_field;
```

Před přidáváním typů a polí je nutné provést kontrolu, zda již v databázi není takováto hodnota zavedena. Jedná se pouze o kontrolní dotaz a není tedy nutné řádné řazení, ale postačí nám zjistit, zda takový záznam existuje nebo ne.

```
SELECT id_field, field FROM field WHERE (field=???);
SELECT id_type, type FROM type WHERE (type=???);
```

Pro vložení nových dat pak budou sloužit jednoduché dotazy, které vloží pouze hodnotu do příslušného textového atributu, protože identifikační čísla se vkládají automaticky pomocí autoincrementu. Také je nutné provést vložení dat do tabulky TYPE_FIELD. Nejdříve tedy dojde k vložení samotného pole nebo typu, kde se vyplní pouze text. V případě, že vkládáme nové pole je nutné pro toto pole a všechny typy vytvořit v tabulce TYPE_FIELD jejich vztah. V případě vložení nového typu je zase nutné vytvořit vztah mezi tímto typem a všemi existujícími poli. K tomu poslouží dotaz vkládající tyto vztahy. V případě, že by MySQL byl schopen zpracovávat poddotazy, bylo by možné provést celou operaci v jediném dotazu. Já však využiji možností programu a budu používat spojení těchto dotazů přes program a použiji pouze dotaz pro vložení jednoho vztahu, který budu programem volat dle potřeby několikrát s různými vstupy v podobě identifikačních čísel typů a polí.

```
INSERT INTO field (id_field, field) VALUES (,???);
INSERT INTO type (id_type, type) VALUES (,???);
INSERT INTO type_field (id_type, id_field, atribut)
      VALUES (???,???,'ost');
```

Při změnách vztahů mezi typy a poli je nutné provést smazání stávajícího záznamu a přidat záznam nový s upravenou hodnotou pro daný vztah. Pro každý vztah je v datové tabulce TYPE_FIELD zadána hodnota odpovídající tomuto vztahu. Je to hodnota REQ pro povinné pole k danému typu, dále OPT, což je doporučené pole a nakonec OST neboli všechna ostatní pole. Při smazání nějakého vztahu je nutné zadat identifikační číslo typu a pole, kterých se tato změna týká. Zadávání atributu je sice zbytečné, neboť pro každou dvojici identifikačních čísel je v databázi pouze jeden záznam, ale je tím možné zamezit smazání záznamu, kterému by neodpovídal tento parametr. Například nelze smazat záznam, o kterém se domnívám, že je povinný a přitom by se jednalo o záznam doporučený. Při vkládání dochází k vložení identifikačních čísel daného typu a pole a parametru pro daný vztah.

```
DELETE FROM type_field
      WHERE (id_type=???) and (id_field=???) and (atribut=???);
INSERT INTO type_field VALUES (???,???,???);
```

Při mazání typů je nejprve nutné zjistit, zda se v druhé části databáze nenacházejí dokumenty, které by byly daného typu. K tomu stačí pouze jednoduchý dotaz, ve kterém v klauzuli WHERE hledám všechny záznamy odkazující na daný typ. V případě, že přistoupím k mazání záznamů týkajících se daného pole je nutné smazat záznamy, jak z tabulky TYPE, tak také z tabulky TYPE_FIELD. V obou těchto dotazech se do klauzule WHERE uvede pouze identifikační číslo mazaného typu.

```
SELECT id FROM dokument WHERE (id_type=???);
DELETE FROM type WHERE (id_type=???);
DELETE FROM type_field WHERE (id_type=???);
```

Pří mazání polí je nutné zjistit, zda nejsou data vyplněná v daném poli u libovolného dokumentu. K tomu je užito jednoduchého dotazu s klauzulí WHERE, která nalezne všechny záznamy vztahující se k identifikačnímu číslu požadovaného pole. Při mazání je nutné smazat záznam z tabulky FIELD i z tabulky TYPE_FIELD, kde jednotlivé záznamy vyberu pomocí klauzule WHERE omezené identifikačním číslem konkrétního pole.

```
SELECT id_dokument FROM data dokument WHERE id_field=???);
DELETE FROM field WHERE (id_field=???);
DELETE FROM type_field WHERE (id_field=???);
```

Toto je celkový přehled všech základních dotazů, které jsou použity pro práci s první částí databáze. Dotazy jsou však pouze zkušební a při použití v programu budou vytvářeny strojově. Je totiž patrné, že mnoho dotazů je stejných pro typy i pole a liší se pouze právě klíčovým slovem TYPE nebo FIELD. Také otazníky bude nutné nahradit konkrétními hodnotami, které budeme získávat z proměnných přenášených v aplikaci.

Pro práci programu bude rovněž potřeba vytvořit různé kontrolní dotazy, které ověří provedení různých operací. Dále bude pravděpodobně nutné vytvořit různé pomocné dotazy pro běh programu, které budou zjišťovat například počty záznamů apod.

4.2.2.2. Dotazy pro druhou část databáze

Pro druhou část databáze je nutné vytvořit dotazy, které budou sloužit k několika účelům. Prvním z nich je vkládání nových dat do databáze, k jejich úpravě a k vyhledávání.

Při přidávání dat je nutné provést přidání záznamu do tabulky DOKUMENT, ve kterém je uveden typ daného dokumentu a také reference. Identifikační číslo je přidáno automaticky pomocí funkce autoincrement, která je zapsána v definici tabulky při jejím vytvoření. Do tabulky je tedy vloženo pouze identifikační číslo typu, který danému dokumentu připadá a hodnota reference.

```
INSERT INTO dokument (id_type, reference) VALUES (???, ???);
```

Následně po vložení nového záznamu o dokumentu je třeba vkládat data o jednotlivých polích vztahujících se k danému dokumentu. V tabulce DATA DOKUMENT se vloží identifikační číslo dokumentu, identifikační číslo pole a text, který se k tomuto poli vztahuje. Toto vložení se tedy provede tolikrát, kolik polí má mít daný dokument vyplněných. To již zajistí samotný program.

```
INSERT INTO data_dokument (id_dokument, id_field, field_text)
                  VALUES (???, ???, ???);
```

Při vyhledávání dat bylo nutné vytvořit dotaz, který umožní vyhledávání dle libovolného pole. Dotaz musí umožňovat nalézt dokument podle jakéhokoliv pole a také dle jakékoliv kombinace těchto polí. Jako výstup dotazu jsem zvolil identifikační číslo dokumentu, referenci, jeho typ, text pole author, text pole title a text pole year. Údaje je tedy nutné z databáze získávat. Proto je nutné při definici tabulky provést spojení polí pomocí identifikačního čísla dokumentu. Dotaz rovněž musí umožňovat řazení dle vybraného atributu. Je pravděpodobné, že se bude vypisovat mnoho záznamů a proto je vhodné použít také klauzuli LIMIT, která vypíše pouze určený počet řádků od požadovaného řádku. Do dotazu je nutné vložit identifikační čísla polí Author, Title a Year pro spojení tabulek. Dále je nutné vložit atribut, podle kterého se budou záznamy řadit a nakonec požadovaný počet vypisovaných řádků a číslo řádku, od kterého se s vypisováním má začít.

```
SELECT dokument.id, dokument.reference, type.type, al_author.field_text as
author, title.field_text as title, year.field_text as year
FROM (((( dokument INNER JOIN type ON
            dokument.id_type = type.id_type)
      LEFT JOIN data_dokument as al_author ON
            dokument.id=author.id_dokument and author.id_field=???)
      LEFT JOIN data_dokument as al_title ON
            dokument.id = title.id_dokument and title.id_field=???)
      LEFT JOIN data_dokument as al_year ON
            dokument.id = year.id_dokument and year.id_field=???)
ORDER BY ???
LIMIT (???,???)
```

Dotaz je bez podmínky WHERE a slouží pouze ke spojení řádků v tabulce dokument a jejich vypsání. Tento dotaz by vypsal informace o všech záznamech v tabulce DATA DOKUMENT je tedy pouze určitou kostrou a je nutné ho dále rozšířit.

Prvním rozšířením bude připojení dalších řádků, které budou odpovídat hledaným polím. Rozšíření se vloží za poslední závorku v klauzuli FROM. Bude také nutné přidat na začátek klauzule otevírací závorku. Do dotazu je nutné vložit název aliasu, který bude pro volání daného pole použit a identifikační číslo tohoto pole. Pro pole author, title a year již není nutné tuto část dotazu vkládat, protože je již obsažena v samotném dotazu, i když se podle těchto polí bude hledat.

```
LEFT JOIN data_all as al_??? ON
      dokument.id = al_???.id_dokument and al_???.id_field=???)
```

Další část dotazu, kterou je nutno přidat je samotná klauzule WHERE, která bude sloužit k zadání podmínek. Pro vyhledávání v dotazu je nutné zadat název aliasu pro daný atribut a jeho obsah, který se má najít. Při hledání je použito funkce LIKE spolu s konstrukcí %???%. Tato konstrukce zajišťuje nalezení jakéhokoliv záznamu, který obsahuje text dosazený za otazníky. Tento text může být v záznamu na jakémkoliv místě, tedy na začátku, uprostřed nebo na konci. Budou nalezeny i záznamy, které obsahují pouze hledaný text. Funkce LIKE stírá také rozdíly mezi velkými a malými písmeny a nalezne daný záznam bez ohledu na to, jak jsou písmena uložena.

Jednotlivé části klauzule se budou spojovat pomocí operátoru AND. Pro každé pole kromě polí Author, Title a Year je nejdříve nutné provést připojení daného řádku pomocí JOINu.

```
WHERE (al_???.field_text LIKE '%???%') and (…)
```

Tento dotaz je již kompletní a lze ho vytvářet i pomocí programu dodáváním vkládaných částí obsahujících klauzuli LEFT JOIN a klauzuli WHERE. Klauzule budou stále stejné a budou se obměňovat pouze místa, kde jsou naznačeny otazníky. Jedná se o název aliasu, který je nutné vložit, identifikační číslo pole odpovídající danému aliasu a nakonec hledaný text.

Při práci s daty je nutné provádět různá zobrazování záznamů. K tomu slouží mnoho dotazů typu SELECT, z nichž se poté programovými funkcemi získávají jednotlivé záznamy a vypisují se na obrazovku.

Při úpravě dat o dokumentu je možné provádět změny v tabulkách DOKUMENT a DATA DOKUMENT.

V první z tabulek lze měnit obsah atributu reference. Pro omezení záznamu, nad kterým se má dotaz provést je použita klauzule WHERE, kde se hledá podle identifikačního čísla dokumentu. Do dotazu se zadává text, který má stávající obsah atributu reference nahradit.

```
UPDATE dokument SET reference=??? WHERE id_dokument=???;
```

V druhé tabulce se může měnit text odpovídající obsahu jednotlivých polí. Pro omezení záznamu, nad kterým se má dotaz provést je použita klauzule WHERE, kde se hledá podle identifikačního čísla pole a identifikačního čísla daného dokumentu. Do dotazu se vkládá text nahrazující obsah atributu field_text.

```
UPDATE data_dokument SET field_text=???
                     WHERE (id_dokument=??? and id_field=???);
```

Pro zobrazení všech informací o nějakém dokumentu je použito několika dotazů. Prvním je dotaz, který získá referenci dané publikace a její typ, v dalším se nalezne obsah příslušného hledaného pole. Všechna data se pak vypíší najednou na obrazovku.

```
SELECT reference FROM dokument WHERE id=???;
SELECT field_text FROM data dokumenty
      WHERE (id_dokument=???) and (id_field=???);
```

Takto vypadají dotazy sloužící pro práci s daty v druhé části databáze. Byly zde uvedeny základní dotazy, ale v samotné aplikaci bude ještě nutné vytvořit mnoho nových či pozměnit stávající tak, aby plně vyhovovaly chodu programu.

5. REALIZACE TŘETÍHO BODU ZADÁNÍ

5.1. Základní seznámení s PHP

5.1.1. Historie

Počátek PHP je datován do roku 1994, kdy Rasmus Lerdorf vytvořil řadu skriptů v Perlu, kterými zjišťoval statistiky o návštěvnosti svých stránek. Pozvolna se lidé začali o tyto skripty zajímat, takže byly později vydány jako balíček "Personal Home Page" tool (původní význam PHP). Vzhledem k velkému zájmu napsal Lerdorf skriptovací jádro společně s jiným nástrojem pro analýzu vstupu z formulářů v HTML, který se nazýval FI (Form Interpreter) neboli PHP/FI či také PHP2.

Později byly tyto nástroje stále častěji využívány pro mnohem komplikovanější věci a vývoj přešel z jednoho člověka na skupinu hlavních programátorů odpovídajících za projekt a jeho organizaci. To byl začátek PHP3. Skupina programátorů (Rasmus Lerdorf, Andi Gutmans, Zeev Suraski, Stig Bakken, Shane Caraveo a Jim Winstead) zlepšila a rozšířila skriptovací jádro a přidala jednoduché API, které umožňuje jiným programátorům volnost v přidávání funkcí do jazyka vytvořením modulů. Syntaxe jazyka byla také zdokonalena konstrukcemi, které budou důvěrně známé lidem, přecházejícím z objektově orientovaných a procedurálních jazyků.

5.1.2. Stávající verze

Stávající verze je verze 5 (PHP5) založená na jádře Zend (detaily najdete na http://www.zend.com). Toto skriptovací jádro bylo navrženo od základu tak, aby bylo jednoduše implementovatelné do různých aplikací. PHP je první aplikací, která používá jádro Zend, ale mělo by být zahrnuto i v jiných aplikacích (např. MySQL). Je možné zaznamenat trend směřující k používání PHP ve více sídlech pro řešení jejich potřeby skriptování. Statistiky ukazují kontinuální zvyšování celkového počtu domén a adres IP, používajících PHP jako interní modul v Apache.

5.1.3. Rozdíl mezi PHP a HTML

Hlavním rozdílem je to, že čistý kód HTML je interpretován prohlížečem a není prováděn na serveru. Napsáním kódu, který je spuštěn na serveru dosáhneme mnoha jiných věcí, které by jinak nebyly možné. Místo přenosu statické stránky HTML k uživateli chceme, aby server provedl některé operace odpovídající našemu kódu PHP. PHP pak vytvoří stránku, která přesně odpovídá situaci. Akce probíhající při zpracování PHP jsou následující:

1. Prohlížeč odešle požadavek na server.
2. Server registruje požadavek prohlížeče.
3. Server nalezne požadovanou stránku na serveru.
4. Server provede instrukce PHP pro vytvoření stránky.
5. Server odešle statickou HTML stránku zpět prohlížeči.
6. Prohlížeč provede zobrazení této statické stránky HTML.

5.1.4. Proč používat PHP

PHP je dnes jedním z nejrozšířenějších programovacích jazyků pro webovské aplikace, který je používán po celém světě. Je založen na základech jazyka C a C++ a tak jej používá řada programátorů, kteří v tomto jazyce programují. Dalším důvodem proč právě PHP je to, že je na Internetu dostupný zdarma a není tak problém s jeho získáním. Tím, že jde o velice rozšířený jazyk, je také jeho podpora velice silná a lze tak na Internetu získat mnoho procedur a objektů, které lze využít ve svém zdrojovém kódu. Jak na Internetu, tak v odborných publikacích lze získat potřebné informace o užívání tohoto programovacího jazyka od základů až po nejsložitější programátorské dovednosti. PHP má také podporu při spolupráci s mnoha databázovými servery a při zpracovávání XML dokumentů. To vše ho předurčuje k širokému užití k programování webovských aplikací.

5.2. Rozbor zadání

Hlavním úkolem bakalářské práce je vytvořit aplikaci umožňující zpracovávání informací o dokumentech uložených ve formátu BibTeX. Tato aplikace musí umožňovat vkládání, opravování a mazání záznamů. Dále musí být schopna vyhledávat záznamy dle jednotlivých polí a musí umožňovat správu typů a polí a jejich vztahů.

Při rozboru zadání je nutné si uvědomit, jak bude provedeno řízení přístupu k databázi a práva uživatelů na práci s daty. Po konzultaci s vedoucím práce bylo rozhodnuto, že touto problematikou se není nutné zabývat a její řešení je součástí jiné práce. Aplikace také nebude pracovat s sessionami a je nutné předávání dat vyřešit jiným způsobem.

5.3. Návrh programu

5.3.1. Práce s databází (objekt Mysql)

Protože aplikace bude provádět mnoho operací nad databází, rozhodl jsem se pro vytvoření objektu, který bude řešit práci s touto databází a umožní přístup pomocí základních metod. Dále umožní vnitřní zpracování případných chybových hlášení a pro další zpracování vrátí již kompletní chybové hlášení, které není nutné dále upravovat. Tím i práce s daty bude jednodušší a názornější. Byl proveden návrh metod a atributů objektu tak, aby byl schopen plně pokrýt všechny požadavky na něj kladené ze strany programu.

Navíc má tento objekt vlastní konfigurační soubor který umožňuje snadné nastavení parametrů databáze a tím i snadné přizpůsobení požadavkům.

5.3.1.1. Konstruktor

V konstruktoru dochází ke kontrole, zda jsou zadány všechny vstupní parametry, kterými jsou:

- $host - jméno počítače, na kterém běží databázový server,
- $databaze - jméno databáze,
- $user - jméno uživatele přistupujícího k databázi,
- $password - heslo uživatele.

Všechny proměnné musí být vyplněny s výjimkou $password. Tyto externí proměnné jsou konstruktorem uloženy jako proměnné objektu a poté proběhne volání vnitřní metody connect.

5.3.1.2. Vnitřní metody

Metoda connect slouží k připojení k databázi. Nejdříve se pomocí parametru volá funkce mysql_pconnect, která do proměnné $link uloží číslo spojení. V případě, že nedojde k připojení je provedeno zpracování chybového hlášení a metoda je ukončena. Poté se zavolá funkce mysql_select_db a ta provede výběr databáze a opět do proměnné $link uloží číslo spojení. Rovněž je provedeno ošetření chybových hlášení.

Metoda free slouží k řádnému ukončení paměti výsledků daného spojení. K tomu slouží funkce mysql_free_result, jejímž parametrem $result. Tato metoda je volána při ukončování spojení.

5.3.1.3. Veřejné metody

První veřejnou metodou je metoda query, která nejdříve zkontroluje, zda je uvolněna paměť, případně ji vyprázdní. Dále zkontroluje, zda není proměnná $query prázdná a použije volání funkce mysql_query, kde se jako parametr vkládá proměnná $query, ve které je dotaz a $link, kde je uloženo číslo spojení. Je provedeno ošetření chybového hlášení a metoda je ukončena.

Metoda getNumberRecords slouží ke spočítání počtu řádků po provedení dotazu. Vstupním parametrem je číslo odkazující na výsledek uložené v proměnné $result. Metoda má ošetřené chybové stavy, kdy není vloženo správné číslo výsledku nebo ve výsledku nejsou žádné řádky. Jinak metoda vrací počet řádků daného dotazu.

Metoda nextRecord slouží k přechodu na následující záznam ve výsledku, který je daný číslem z proměnné $result. Jsou zde ošetřeny chybové stavy, kdy není vloženo správné číslo výsledku nebo není možné na další záznam přejít. Jinak je proveden přechod na následující řádek.

5.3.1.4. Atributy objektu

Pro přístup k datům je použito atributu record[X], kde x udává o kolikátý atribut daného záznamu se jedná. Pro posun mezi záznamy je použito metody nextRecord.

5.3.2. Zpracování chybových hlášení (objekt ErrorHandler)

Tento objekt je určen hlavně pro potřeby tvůrce aplikace a pro administrátora. Má mnoho funkcí výpisu chybových hlášení, varování i poznámek. Umožňuje velmi snadnou detekci případných potíží a urychluje tak programátorskou práci i práci administrátora aplikace.

5.3.2.1. Konstruktor

Konstruktor slouží k nastavení mnoha parametrů, které ovlivňují chod celého systému zpracování chybových hlášení.

5.3.2.2. Veřejné metody

Většina veřejných metod je určena k vypisování chybových hlášení, upozornění nebo poznámek na různé úrovni. Umožňuje jednak výpis do prohlížeče a to metodami setViewNoticeDebugOnBrowser a setDebugOnBrowser, dále výpis do log souborů pomocí metody setViewNoticeWriteToLogFile a také zasílání těchto hlášení na email pomocí metody setViewNoticeEmail.

5.3.3. Struktura programu

Program pracuje tak, že se spustí základní soubor, do kterého je dle proměnné načten příslušný rozšiřující soubor s odpovídající částí programu, která zpracovává konkrétní situaci. Tyto soubory se dotahují ze samostatného adresáře /include. V konkrétní načtené části programu jsou použity funkce, které jsou volány z knihoven, uložených v samostatném adresáři /lib. Dále je použit konfigurační soubor, v němž lze nastavit základní parametry pro běh programu.

5.3.3.1. Základní soubor

Základní soubor s názvem index.php se načítá vždy a je ve většině prohlížečů nastaven jako implicitní pro zpracování při příchodu na www adresu.

První k čemu dojde při startu tohoto souboru, je vyprázdnění bufferu pro výstup a jeho spuštění. Poté dochází k zaznamenávání všech výstupů z programu a ty se ukládají do bufferu a klientovi jsou zaslány až po žádosti ze strany programu. To také umožňuje v případě vážné chyby buffer vymazat a místo stávajícího kódu zaslat klientovi kód pro zobrazení chybového hlášení. Dalším krokem při zpracovávání základního souboru je načtení konfiguračního souboru config.inc.php a knihovny pro zpracovávání požadavků. Poté již dochází k větvení programu a načítání příslušných vkládaných souborů dle vstupních parametrů. K načítání souborů je užito php funkce require. Posledním krokem je vytištění patičky stránky pomocí funkce patička z knihovny main.lib.php. Tím je běh tohoto souboru ukončen a čeká se na odezvu klienta, který zašle požadavek opět na tento, soubor ovšem s jinými vstupními parametry. Tento postup se opakuje až do ukončení práce s programem.

5.3.3.2. Vkládané soubory

Všechny vkládané soubory jsou umístěny v adresáři /include a mají příponu *.inc.php. Při každém průchodu základním souborem je vložen pouze jeden tento soubor, který provede odezvu na klientovy požadavky a na výstup zašle výsledky.

Úvodní soubor

V případě, že žádný ze vstupních parametrů neodpovídá podmínkám při větvení v základním souboru, dojde k načtení úvodního souboru. Tento soubor provede funkci hlavička z knihovny main a následně zobrazí hlavičku stránky s navigačními tlačítky a úvodním textem. Poté je zobrazen nadpis aplikace a vkládaný soubor je ukončen.

Přidání dokumentu

Prvním souborem, který se zobrazí při přidávání dokumentu je soubor includePubl.inc.php. V případě, že nebyl potvrzen formulář a proměnná action není definována spustí se první část souboru. Ta zobrazí na obrazovce pouze výběrové pole obsahující všechny typy a potvrzovací tlačítka. Výběrové pole je zobrazeno funkcí showSelect s parametrem definujícím tabulku, ze které se mají vybrat záznamy do tohoto pole.¨

V případě, že byl formulář odeslán a proměnná action je definována, pokračuje se zobrazením typu dokumentu a zobrazením tří sloupců, kde v prvním z nich jsou zobrazena textová pole pro povinná pole, ve druhém jsou textová pole pro doporučená pole a ve třetím sloupci jsou textová pole pro volitelná pole. Jednotlivá textová pole jsou do sloupců zobrazena funkcí showFieldPubl, která má jako vstupní parametr text definující, zda se jedná o povinné nebo doporučené pole, dále identifikační číslo typu dokumentu, identifikační číslo dokumentu. Nakonec dojde k zobrazení potvrzovacích tlačítek. Po vyplnění a odeslání tohoto formuláře dojde ke kontrole povinných polí. Pokud nejsou vyplněna všechna, dojde k opětovnému zobrazení tohoto formuláře s předvyplněnými hodnotami.

Po úspěšném vyplnění dojde k zobrazení formuláře pro vyplnění reference, která je vytvořena dle standardních parametrů pro Bibtex. Referenci lze volně měnit. Po odeslání formuláře dojde ke kontrole jedinečnosti reference. Pokud není reference jedinečná dojde k opětovnému zobrazení tohoto formuláře spolu s informací, že reference není jedinečná a je třeba ji změnit.

Nakonec dojde k zobrazení všech vyplněných hodnot a soubor je ukončen.

Hledání dokumentu

Pro samotné hledání dokumentu je použito pouze jednoho souboru s názvem findPubl.inc.php, který slouží k zadání parametrů pro hledání a k zobrazení výsledků. Není-li formulář odeslán, a tím není definovaná proměnná action, dojde k zobrazení hlavičky stránky a poté se zobrazí do tří sloupců textová pole s názvy pro všechna pole. Pod nimi se zobrazí potvrzovací tlačítka. V případě, že již došlo k odeslání formuláře a proměnná action je definována, zobrazí se hlavička stránky s navigačními tlačítky. Následuje kód sloužící ke zpracování tlačítek pro listování ve výsledku (viz. dále), kde se do proměnné aktualni_stranka nastavuje hodnota po stisknutí některého z tlačítek pro listování. Následuje funkce showFindPubl, která na obrazovku vypíše výstup dle zadaných parametrů hledání. Vstupním parametrem této funkce je pole s hledanými řetězci, název atributu, dle kterého mají být záznamy seřazeny, číslo aktuální zobrazované stránky a počet zobrazovaných řádků. Poté dojde k zobrazení navigačních tlačítek pro listování ve stránkách a tím je soubor ukončen.

Přehled dokumentu

Po vyhledání publikace je možné provést její prohlížení a úpravy. K tomu je určen soubor updatePubl.inc.php, který zobrazí hlavičku stránky a poté zobrazí danou publikaci dle předaného parametru. Zobrazí se typ dokumentu reference a tři sloupce, kde v prvním z nich jsou zobrazena textová pole pro povinná pole, ve druhém jsou textová pole pro doporučená pole a ve třetím sloupci jsou všechna ostatní pole, která byla vyplněna. K tomu slouží funkce showFieldPubl jejímiž vstupními parametry jsou, text definující zda se jedná o povinné, doporučené nebo ostatní pole, identifikační číslo typu a identifikační číslo dokumentu. Je zde umožněno přejít tlačítkem k úpravě dané publikace.

Pro úpravu publikace je použita strukturu jako soubory pro přidávání nových dokumentů s tím rozdílem, že se nevybírá typ daného dokumentu a referenci lze znenit současně s ostatními parametry. Proto není nutné soubor dále popisovat a postačí odkázat na soubor includePubl.inc.php.

Číselníky

Pro práci s číselníky je použita skupina souborů s názvem ciselnik*.inc.php. Všechny soubory nejprve načtou knihovny pro práci s databází, pro zobrazování a pro editaci. Následně se zobrazí hlavička dokumentu a dojde k zaslání hlavičky stránky s logem a navigačními tlačítky na výstup.

Prvním souborem je ciselnik.inc.php. Tento soubor je prvním souborem, který je použit. Je zobrazen select, v němž jsou z databáze dotaženy všechny typy a k aktuálnímu typu jsou zobrazena povinná a doporučená pole. K zobrazení typu je použito funkce showSelect se vstupními parametry, které definují z jaké tabulky se mají data zobrazit a identifikační číslo aktuálního typu. Zobrazení polí se provede funkcí showField s parametrem definujícím o jaké pole má jít a identifikačním číslem typu pro daná pole. Tím je soubor ukončen.

Po volbě navigačního tlačítka Přidat typ dojde k načtení souboru ciselnikIncludeType.inc.php. Není-li definován parametr action, dojde k zobrazení pole pro vkládání textu nového typu a k zobrazení potvrzovacích tlačítek. K zobrazení vkládacího pole je použita funkce showPridej, která má jako

vstupní parametr název vkládaného atributu, defaultní hodnotu a jméno tabulky, do které se má hodnota vložit. V případě, že je definována hodnota action, neboli došlo k odeslání formuláře, proběhne kontrola vkládaného textu. Nejdříve se php funkcí ereg zkontroluje, zda text obsahuje přípustné znaky. Dále se kontroluje, zda již typ s daným textem neexistuje. K tomu slouží funkce id volaná s parametry, kterými jsou název vkládaného typu a jméno tabulky, ve které se má hledání provádět. V případě, že ano, je vypsáno chybové hlášení a je-li vše v pořádku, dojde k přidání typu do tabulky funkcí pridej s parametry jméno tabulky a textem, který se má vložit. Tím soubor končí.

Další vkládaný soubor slouží k přidávání nového pole se jmenuje ciselnikIncludeField.inc.php a lze ho vyvolat navigačním tlačítkem Přidat pole. Tento soubor je velmi podobný souboru pro vkládání typu. Není-li definována proměnná action dojde k zobrazení textového pole sloužícího pro vkládání názvu pole a k zobrazení potvrzovacích tlačítek. Používá se i stejná funkce showPridej s rozdílnými parametry. V případě, že text byl již odeslán potvrzovacím tlačítkem a tím je definována proměnná action, dojde ke kontrole obsahu textového pole php funkcí ereg. Následně dojde ke kontrole, zda již není takovéto pole v databázi zavedeno pomocí funkce id, která má jako parametr název nového pole a jméno tabulky, ve které má dojít ke kontrole. Oba stavy jsou ošetřeny chybovým hlášením a v případě, že je vše v pořádku, dojde k přidání daného pole do databáze a soubor končí.

Pro odebrání typu slouží soubor ciselnikDeleteType.inc.php vyvolaný navigačním tlačítkem Odstranit typ. Tento soubor v případě nedefinované proměnné action zobrazí výběrové pole naplněné všemi typy a pod ním se zobrazí potvrzovací tlačítka. K zobrazení výběrového pole je použito funkce showSelect s parametrem definujícím název výběrového pole a jménem tabulky, ze které se mají požadovaná data pro naplnění získat. V případě, že došlo k odeslání a je definována proměnná action, dojde ke kontrole, zda je možné aktuální typ smazat. K tomu je použito funkce controlTypeDelete, která zjistí, zda není v databázi nějaký dokument, který by byl daného typu. V případě, že je vše v pořádku, provede se smazání typu funkcí deleteType s parametrem, kterým je identifikační číslo daného typu. Poté se zobrazí případné chybové hlášení a soubor je ukončen.

Volba navigačního tlačítka Odstranit pole má za následek natažení souboru pro odstranění pole ciselnikDeleteField.inc.php. Není-li definována proměnná action, zobrazí se, stejně jako při odstraňování typu výběrové pole se všemi poli, potvrzovací tlačítka. K zobrazení výběrového pole je použito funkce showSelect s parametrem názvu výběrového pole a názvu tabulky, ze které se mají pole dotáhnout. Je-li proměnná action definována, dojde ke kontrole, zda lze vybrané pole smazat pomocí funkce controlFieldDelete, do které jako parametr vstupuje identifikační číslo pole. Funkce zjistí, zda daný typ není používán jako povinný u nějakého pole a zda není v databázi dokument s tímto vyplněným polem. Je-li vše v pořádku, dojde ke smazání pole funkcí deleteField s parametrem definujícím identifikační číslo pole. Poté se zobrazí případné chybové hlášení a soubor končí.

Posledním volaným souborem pro práci s číselníky je soubor ciselnik_edit.inc.php. Tento soubor je nahrán po volbě navigačního tlačítka Opravit vztahy a slouží k provádění úprav vztahů mezi typy a povinnými či doporučenými vztahy. Nedošlo-li k odeslání stránky a není-li definována proměnná action, zobrazí se tabulka se třemi sloupci. V prvním z nich je výběrové pole se všemi typy, které je zobrazeno funkcí zobrazSelect s parametrem definujícím tabulku, ze které se má toto pole naplnit. Druhý a třetí sloupec bude obsahovat zaškrtávací boxy a názvy všech polí. V druhém budou zaškrtnuty ty boxy, které budou odpovídat povinným polím daného typu a ve třetím ty, které budou odpovídat doporučeným polím aktuálního typu. Oba sloupce budou naplněny funkcí showFieldEdit, která má dva parametry. Prvním z nich je text definující, zda se jedná o povinné nebo doporučené pole a druhým je identifikační číslo typu. Nakonec se zobrazí potvrzovací tlačítka. Došlo-li již k odeslání formuláře a je definována proměnná action, dojde ke kontrole provedených změn, k čemuž je použito vnořeného cyklu for, který ověří, zda není jedno a to samé pole použito jako povinné i doporučené pole zároveň. V případě, že je vše v pořádku, dojde ke smazání povinných a doporučených polí, která jimi již nemají být. K tomu slouží funkce deleteTypeField, která má za vstupní parametr text definující, zda se jedná o povinné nebo doporučené pole, dále identifikační číslo typu, kterého se dané změny týkají a seznam všech původních a nových povinných nebo doporučených polí. Poté dojde naopak k přidaní všech nových polí, k čemuž je použito obdobné funkce insertTypeField se stejnými parametry. Následuje vypsání případných chybových hlášení a soubor je ukončen.

5.3.3.3. Knihovny

Program využívá další vkládané soubory jako své knihovny. Jde o skupiny funkcí, které jsou programem volány a lze je proto nazývat knihovnami. Jde o soubory zobraz.lib.php, edit.lib.php a funkce.lib.php. Tyto knihovny jsou rozděleny podle své funkce na zobrazování výstupů na obrazovku, práci s daty a základní funkce. Všechny tři knihovny před svým spuštěním ověří, zda již nedošlo k jejich natažení a minimalizují tak duplicitní natahování do systému a jeho zbytečné přetěžování. K tomu je použito globální proměnné, ve které je zaznamenáno, zda již došlo k natažení do systému.

Knihovna EDIT

První funkcí této knihovny je idPrvni, která zjistí první identifikační číslo z tabulky definované vstupním parametrem funkce. Další funkce id vrací identifikační číslo záznamu z tabulky definované vstupním parametrem, který splňuje podmínku danou druhým ze vstupních parametrů.

Následuje funkce insert, která slouží k přidání nového záznamu do tabulek polí a typů. Vstupním parametrem je zde jméno tabulky a vkládaný text. Funkce nejprve přidá záznam do příslušné tabulky a poté zjistí jeho identifikační číslo. Dále vloží do tabulky TYPE_FIELD potřebné záznamy definující vztah mezi novým typem, resp. polem a všemi ostatními poli, resp. typy.

Funkce deleteTypeField slouží ke smazání všech záznamů odkazujících na již neplatné vztahy mezi typy a poli. Vstupním parametrem funkce je text definující, zda se jedná o povinná nebo doporučená pole, dále identifikační číslo typu a pole se seznamem identifikačních čísel polí před změnou a po změně. Funkce v cyklu for ověřuje, zda dojde ke shodě mezi identifikačními čísli polí před změnou a po změně. V případě, že k této shodě nedojde a identifikační číslo před změnou nemá svého oponenta mezi identifikačními čísli polí po změně, dojde ke smazání tohoto záznamu a přidání záznamu o tom, že dané pole patří mezi ostatní pole. Funkce insertTypeField má stejné vstupní parametry a pracuje také stejně s tím rozdílem, že nejdříve dochází ke smazání záznamu o tom že pole patří mezi ostatní a poté se přidá záznam definující požadující vztah mezi typem a polem.

Funkce deleteType smaže záznam z tabulky TYPE a tabulky vztahů TYPE_FIELD s identifikačním číslem, které je vstupním parametrem funkce. Druhá funkce deleteField pracuje stejně s tím, že záznamy maže z tabulky FIELD a TYPE_FIELD s identifikačním číslem, které je opět vstupním parametrem funkce.

Následují dvě funkce k ověření možnosti smazání záznamu o poli nebo typu. Tyto funkce se jmenují controlTypeDelete a controlFieldDelete, jejichž vstupním parametrem je identifikační číslo typu, resp. pole. Tyto funkce ověřují, zda na dané záznamy nejsou v tabulkách nějaké odkazy, které by se v případě smazání staly nefunkční a smazání záznamů by tak narušilo integritu databáze.

Funkce createPubl se vstupním parametrem definujícím identifikační číslo typu daného dokumentu. Provede vytvoření nového záznamu o dokumentu a zjištění jeho identifikačního čísla. Funkce getTypPubl vypíše identifikační číslo typu dokumentu dle identifikačního čísla dokumentu, které je vstupním parametrem.

Další funkce setFieldPubl má jako vstupní parametr název vkládaného pole, jeho obsah a identifikační číslo dokumentu. Funkce nejprve ověří, zda se v tabulce DOKUMENT již nachází záznam k danému poli a poté provede vložení, resp. upravení záznamu dle vložených hodnot.

Pro zpracování reference je použito funkce insertReference, která má vstupní parametry definující text vkládané reference a identifikační číslo dokumentu. Funkce provede upravení hodnoty reference záznamu v tabulce DOKUMENT daného identifikačním číslem dokumentu.

Funkce deletePubl má vstupní parametr definující identifikační číslo dokumentu a provede smazání všech záznamů z tabulek DOKUMENT a DATA DOKUMENT, které náleží k dokumentu s daným identifikačním číslem.

Knihovna VIEW

První funkce této knihovny je showSelect ze vstupními parametry definujícími tabulku, ze které mají být získány hodnoty pro zobrazení, identifikačním číslem záznamu, který má být aktuální, pravdivostní hodnotou definující, zda může být hodnota měněna a textem pro klauzuli WHERE vkládanou do dotazu pro výběr záznamů. Nejprve dojde k zobrazení HTML tagu select, který se

naplní záznamy z dotazu, který je definován vstupními parametry. V případě, že je zapsána aktuální hodnota identifikačního čísla záznamu, je tento záznam označen jako selected a dle definované pravdivostní hodnoty je případně zobrazeno disable na daný select.

Další funkce showFieldEdit má vstupní parametry definující text označující zda se jedná o pole povinná nebo doporučená a identifikační číslo typu. Nejprve dojde k nalezení všech povinných, resp. doporučených záznamů pro daný typ z databáze, dále k získání všech polí z databáze, které se spočítají a dojde k jejich zobrazení ve dvou sloupcích. K tomu je použito HTML tagu INPUT s parametrem checkbox. V případě, že se jedná o povinné, resp. doporučené pole, je toto pole zaškrtnuto pomocí parametru checked.

Funkce ShowFieldPubl slouží k zobrazení textových polí pro všechny povinné, resp. doporučené záznamy daného typu. V případě, že se mají zobrazit ostatní pole funkce, vytvoří výběrové pole, ve kterém jsou všechna pole, která nejsou povinná ani doporučená pro daný typ. Identifikační číslo typu je vstupním parametrem funkce, stejně jako text definující, zda se jedná o pole povinná, doporučená nebo ostatní. Jde-li o povinná nebo doporučená pole, dojde k zobrazení textového pole s názvem daným jménem daného pole. Jde-li o ostatní typy zobrazí se HTML tag select a je naplněn všemi získanými poli pro daný typ. Je-li definován parametr funkce pro zablokování pole, dojde k němu parametrem tagu disable.

Další funkce showReferenc slouží k zobrazení textového pole pro vkládání reference. Vstupním parametrem funkce je identifikační číslo dokumentu a pravdivostní hodnota pro možnost zablokování pole. Tato funkce zjistí hodnoty polí autor, název publikace a rok, příslušně je upraví a tuto hodnotu naplní do textového pole. Je-li definován parametr funkce pro zablokování pole dojde k němu parametrem tagu disable.

Poslední funkce showFindPubl má jako vstupní parametr seznam hledaných řetězců, dále počet řádků, které mají být zobrazeny a aktuální zobrazovanou stránku. Funkce provede vytvoření dotazu hledajícího záznamy v databázi podle požadavků ze vstupu a následně provede jejich zobrazení na obrazovku.

6. JAK PRACOVAT S PROGRAMEM

6.1. Základní zobrazení na obrazovce

Základní zobrazení na obrazovce má v horní části dekorativní hlavička s nadpisem. V levé části se nachází navigační tlaačítka. Poté následuje část určená pro práci s daty, kde se zobrazují výběrová pole, zaškrtávají boxy, textová pole apod. Poté jsou dle potřeby zobrazena potvrzovací tlačítka Potvrď a Zruš a celá stránka je ukončena patičkou.

6.2. Úvodní obrazovka

Po spuštění www stránky se zobrazí základní obrazovka s šesti navigačními tlačítky. Tato tlačítka slouží k volbě, přidání dokumentu, hledání dokumentu nebo pracování s číselníky polí a typů. Další tlačítka jsou určena pro zobrazení nápovědy a dokumentace. Je zobrazen název aplikace a krátký popisný text.

6.3. Přidání dokumentu

Po volbě navigačního tlačítka Přidej publikaci z úvodní obrazovky se zobrazí na obrazovce výběrové pole, ve kterém je možné vybrat jeden z typů. Po výběru typu je nutné stisknout tlačítko Potvrď.

Vybraný typ se zobrazí a pod ním jsou tři sloupce. V prvním z nich jsou zobrazena textová pole pro povinná pole, ve druhém jsou textová pole pro doporučená pole a ve třetím sloupci jsou textová pole pro ostatní pole. Pro zdárné pokračování je nutné vyplnit všechna povinná pole. Pokud zůstanou některá prázdná dojde k zobrazení varovného hlášení a zobrazí se znovu tento formulář s s již vyplněnými hodnotami, ve kterém je nutné zbývající povinná pole doplnit. K odeslání formuláře slouží tlačítko Potvrď a tlačítkem Zruš lze formulář vynulovat.

Po stisknutí tlačítka dojde ke kontrole vyplnění všech povinných polí a v případě, že nejsou vyplněna, zobrazí se chybové hlášení a poté je proveden návrat do přehledu. Pokud je vše v pořádku, zobrazí se textové pole pro vložení reference daného dokumentu. V tomto textovém poli je přednastavena hodnota složená z prvních znaků jména autora, názvu dokumentu, dvojtečka a posledního dvojčíslí z roku vydání. Tuto referenci je možné změnit a po této změně je nutné provést potvrzení tlačítkem Potvrď. Po potvrzení dojde ke kontrole, zda je daná reference jedinečná. Není-li tomu tak, dojde k zobrazení chybového hlášení a poté je nutné referenci změnit.

Je-li vše v pořádku, dojde k zobrazení úplného přehledu dokumentu, kde je vidět typ dokumentu, náhled na všechna povinná a doporučená pole a na všechna ostatní vyplněná pole. Dále je zobrazena také reference dokumentu.

6.4. Hledání dokumentu

Po stisknutí navigačního tlačítka Vyhledej publikace na úvodní obrazovce se zobrazí tři sloupce textových polí, která odpovídají všem polím. Do těchto textových polí je možné napsat hledaný text. Pro nalezení nějakého dokumentu je nutné, aby tento dokument obsahoval všechny uvedené hledané řetězce. Po vyplnění všech hledaných řetězců do příslušných polí, je nutné stisknout potvrzovací tlačítko Potvrď.

Po odeslání hledaných řetězců se na obrazovce zobrazí výsledky hledání. Ty jsou zobrazeny v tabulce o pěti sloupcích, kde v prvním sloupci je zobrazena reference dokumentu, ve druhém je jeho typ, ve třetím se nachází autor dokumentu, ve čtvrtém je název dokumentu a konečně v pátém sloupci je rok vydání dokumentu. Záznamy jsou seřazeny dle reference, ale řazení lze změnit kliknutím na vybraný atribut pro řazení, kterým může být jakýkoliv z pěti uvedených údajů. V případě, že je počet nalezených dokumentů velký, aktivuje se možnost listování ve výsledku hledání pomocí tlačítek zobrazených pod přehledovou tabulkou. Základní nastavení počtu záznamů na jedné obrazovce je patnáct, ale je možné ho změnit vyplněním libovolného čísla do textového pole mezi tlačítky pro listování. V případě zapsání nového čísla počtu zobrazovaných záznamů, je nutné stisknout tlačítko Aktualizuj, které provede překreslení obrazovky dle požadavku. Pro prohlížení detailu každého dokumentu je možné kliknout na zvýrazněnou referenci.

Po výběru dokumentu pro prohlížení detailu dojde k zobrazení úplného přehledu dokumentu, kde je zobrazen typ dokumentu, náhled na všechna povinná a doporučená pole a na všechna ostatní vyplněná pole. Dále je zobrazena také reference dokumentu. Daný záznam lze smazat stisknutím navigačního tlačítka Odeber nebo přejít k úpravě dokumentu pomocí navigačního tlačítka Upravit.

Při odebírání dokumentu se program dotáže, zda má daný dokument opravdu smazat a čeká na potvrzení tlačítkem Potvrď.

V případě, že je zvoleno tlačítko Upravit v přehledu o dokumentu dojde k zobrazení úplného přehledu dokumentu, který umožňuje volit jednotlivá pole a měnit jejich obsah. Tento postup je stejný jako v případě vkládání nového dokumentu včetně přiřazování reference. Po úpravě všech polí a zadání reference, dojde opět k zobrazení úplného přehledu dokumentu.

6.5. Práce s číselníky

Po stisknutí navigačního tlačítka Číselník na úvodní obrazovce se zobrazí obrazovka, na které je výběrové pole obsahující typy a pod ním přehled všech povinných a doporučených polí k aktuálnímu vybranému typu. Tento typ lze měnit ve výběrovém poli a prohlížet tak povinná a doporučená pole všech typů. Na stránce jsou zobrazena navigační tlačítka sloužící pro přidávání a mazání typů i polí a k úpravě vztahů mezi nimi.

Při přidávání nového typu se na obrazovce zobrazí textové pole, do kterého je nutné zadat název nového typu.. Celková délka názvu typu je omezena na patnáct znaků a musí vždy začínat znakem @. Do textového pole je možné vkládat pouze alfabetické znaky. Po vyplnění názvu typu je třeba stisknout potvrzovací tlačítko Potvrď. Poté je provedena kontrola vyplněného textu a jeho uložení. Při přidávání nového pole je postup stejný, jen se liší povolené znaky. První písmeno názvu pole musí být velké a ostatní malé. Po vložení nového pole nebo typu se opět zobrazí úvodní obrazovka pro práci s číselníky.

Při volbě mazání typu se na obrazovce zobrazí výběrové pole, ve kterém jsou obsaženy všechny typy. Po vybrání jednoho z nich je nutné stisknout potvrzovací tlačítko. Po jeho stisknutí se provede kontrola, zda na daný typ neodkazuje některý záznam. V případě, že je vše v pořádku, provede se smazání typu a je zobrazena úvodní obrazovka pro práci s číselníky. V případě, že daný typ nelze smazat je na obrazovce zobrazeno chybové hlášení a výběr lze opakovat.

Při mazání pole je postup podobný. Při kontrole je zjištěno, zda se v databázi nenachází záznam s vyplněným daným polem nebo zda není toto pole určeno u některého typu jako povinné.

Poslední funkcí programu při práci s číselníky je úprava vztahů mezi typy a poli. Po zvolení navigačního tlačítka se vlevo zobrazí výběrové pole se všemi typy a vpravo jsou zobrazeny dva sloupce zaškrtávacích polí s popisem názvů polí. V případě, že je konkrétní pole u aktuálního typu povinné či doporučené je tento zaškrtávací box zatržen. Úpravy vztahů se provádějí zaškrtáváním jednotlivých boxů. Omezující podmínkou je, že se žádné pole nesmí zároveň vyskytovat jako povinné i doporučené. Tato kontrola se provádí po potvrzení změn tlačítkem Potvrď. Je-li vše v pořádku, jsou dané změny provedeny a na obrazovce se zobrazí úvodní stránka práce s číselníkem, jinak je dokument navrácen do původního stavu a je nutné zadávání opakovat.

6.6. Nápověda

V této části se nachází krátká nápověda jak s aplikací pracovat. Její obsah je podobný této kapitole bakalářské práce a slouží k ulehčení práce přímo v aplikaci bez nutnosti čtení této práce.

6.7. Dokumentace

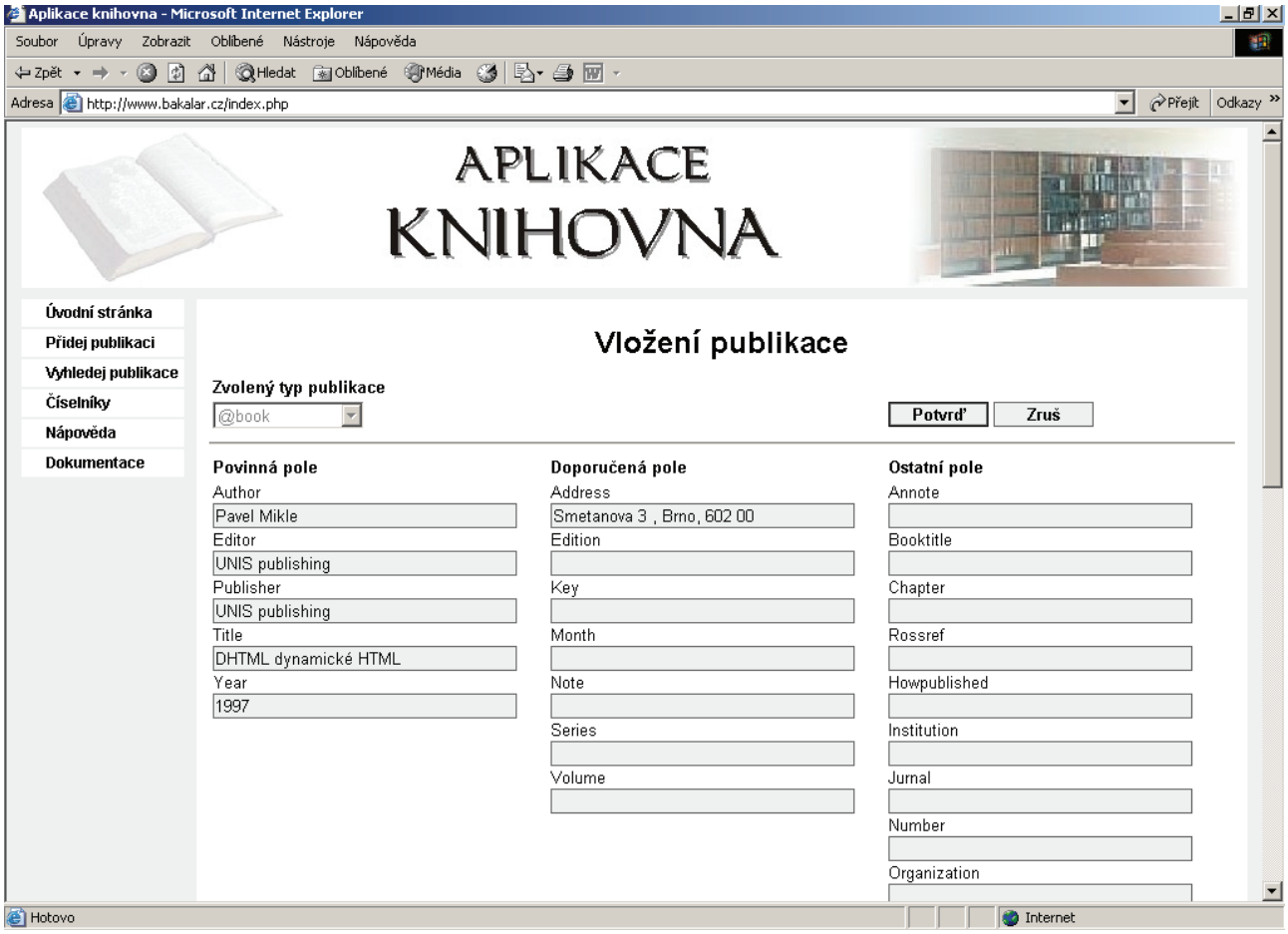
V této části je kompletní dokumentace všech zdrojových souborů včetně jejich plného znění. Tato dokumentace byla vytvořena s využitím produktu phpDocumentor verze 1.3.0RC6.

6.8. Ukončení práce

Práci s programem lze ukončit nejlépe odhlášením od www stránek z úvodní obrazovky celého programu. V případě odhlášení na jakékoliv jiné stránce může dojít k tomu, že poslední prováděné změny nebudou zaznamenány, protože nedošlo k jejich řádnému potvrzení.

6.9. Vzorová obrazovka aplikace

Pro ilustraci grafického rozhraní programu a přiblížení jednoduchosti celého ovládání přikládám sejmutou obrazovku z částí atributů vkládané publikace.



7. ZÁVĚR

Ve své bakalářské práci jsem si ověřil úplný postup tvorby aplikace pracující s databází MySQL s interfacem naprogramovaným pomocí jazyka PHP.

Nejprve jsem navrhl datovou strukturu pro danou aplikaci a provedl úplnou analýzu tohoto návrhu spojenou s konzultacemi s vedoucím práce. Tento návrh spolu s analýzou pro mne znamenal získávání důležitých poznatků spojených s problematikou provozovanou v praxi, neboť konzultace s vedoucím práce lze do značné míry chápat jako konzultace s potencionálními zákazníky.

Při výběru produktů pro vytvoření aplikace jsem byl omezen striktními požadavky ze strany zadavatele, tak jak tomu bývá ve většině případů v praxi, kde jsou limitujícím faktorem obvykle finanční prostředky uvolněné na daný projekt. Toto striktní omezení mělo zásadní vliv jak na zpracování datové struktury, tak její změny a zároveň si vynutilo hledání nových postupů pro realizaci řešení.

Tvorba samotného programového vybavení aplikace byla pro mě přínosná nejen v získání mnoha zkušeností použitelných v praxi, ale zejména získáním ucelené představy o návrhu a tvorbě poměrně složité aplikace a o časové i znalostní náročnosti takovéhoto projektu.

Vytvořená aplikace je již plně funkční pro použití na Internetu, zároveň však poskytuje další prostor pro její rozšíření. Jako příklady je možno uvést vylepšené vyhledávání dokumentu pomocí fulltextového vyhledávání nebo naprogramování tiskových výstupů aplikace a strojové zpracování dat získávaných z jiných datových zdrojů. Na základě uvedených skutečností lze konstatovat, že navržené projektové řešení aplikace není konečné a je možno ho přijmout jako podnět k další diskusi otevírající prostor pro její rozšíření, resp. doplnění další funkčnosti .

Literatura

[1] David Axmark, Michael (Monty) Widenius, Jeremy Cole, and Paul DuBois: MySQL 5.0 Reference Manual, http://www.mysql.com/documentation/index.html, 26 Sep. 2001

[2] Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Andrei Zmievski, Jouni Ahto: Manual PHP, http://www.php.net/docs.php, 21 Jul. 2002

[3] George Greenwade: Help On BibTeX Version 1.0, http://www.cgg.cvut.cz, 12 Apr. 1994

[4] Pavel Mikle, DHTML dynamické HTML, Unis publishing 1997, ISBN 80-86097-09-9

Software

[1] MySQL Database Servers and Clients 5.0, http://www.mysql.com

[2] PHP 5.1.4, http://www.php.net

[3] PhpMyAdmin 2.8.1, http://wwwphpmyadmin.net

[4] Textový editor PSPad 4.5.1 (2207), http://www.pspad.com/cz

[5] phpDocumentor – 1.3.0RC6, http://wwwphpdoc.org