UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Tvorba WWW aplikace s využitím relační databáze pro DVD půjčovnu

Tomáš Linhart

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš LINHART**
Osobní číslo:	**I07703**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**WWW aplikace s využitím relační databáze pro DVD půjčovnu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem praktické části je realizace informačního systému pro půjčovnu DVD s využitím relační databáze.
Teoretická část se bude zabývat návrhem vlastní databáze s důrazem na normalizaci tabulek a porovnáním možností databází Oracle a MySQL.
Aplikace musí umožnit celkovou správu a administraci systému, bude pokrývat registraci uživatelů a jejich přístup dle práv, evidenci filmů, herců, režisérů, hudebních skupin a jejich alb a též jejich vyhledávání, dále pak evidenci výpůjček a k nim generovaných upomínek, evidenci aktuálních novinek pro uživatele a evidenci dotazů uživatelů .

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

***Castagnetto, J. a kol. Programujeme PHP profesionálně. Computer Press, 2004.**
***Oppel, A. Databáze bez předchozích znalostí. Computer Press, 2006.**
***Lacko, L. Oracle, správa, programování a použití databázového systému. Computer Press, 2007.**
***Kofler, M. Mistrovství v MySQL 5 Kompletní průvodce webového vývojáře. Computer Press, 2007.**

Vedoucí bakalářské práce: **RNDr. Iva Rulicová**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2010**
Termín odevzdání bakalářské práce: **14. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlášení autora

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1. autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 14. 5. 2010 Tomáš Linhart

Anotace

Bakalářská práce se zabývá tvorbou a popisem webové aplikace s využitím správně navrhnuté a normalizované relační databáze. Aplikace využívá databázový systém Oracle a je naprogramována v programovacím jazyku PHP. Dále se zabývá odlišnými postupy při použití databázového systému MySql. Aplikace dále využívá technologie jako JavaScript, XML a CSS.

Klíčová slova

film, hudba, půjčovna, dvd, databáze, oracle, mysql, php, javasript

Title

Creation of WWW application with usage relational database for DVD rental.

Annotation

Diploma paper concern with design and description web application with using right designed and normalized relational database. Application using database system Oracle and is programmed in PHP language. Further it concern alternative methods by using database system MySql. Application further using technologies like JavaScript, XML and CSS.

Keywords

movie, music, rental, dvd, database, oracle, mysql, php, javasript

Poděkování

Chtěl bych poděkovat RNDr. Ivě Rulicové za odborné vedení a pomoc při tvorbě této bakalářské práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých odborných výrazů

Odborný výraz	Popis výrazu
Apache	Webový server.
CAPTCHA	Test, který se snaží rozlišit člověka od robotů.
CSS	Kaskádové styly.
HTML	Hypertextový značkovací jazyk.
JavaScript	Skriptovací jazyk na straně klienta.
MVC	Softwarová architektura.
NULL	Prázdná hodnota – (nikoliv 0, spíše hodnota „nic“).
PHP	Skriptovací jazyk na straně serveru.
Rich Picture	Diagram návrhu systému.
SPAM	Nevyžádaná zpráva.
SQL	Strukturovaný dotazovací jazyk.
TRIGGER	Procedura ochuzená o některé možnosti automaticky spuštěná nastavenou událostí.
WYSIWYG	Dokument na obrazovce bude stejný jako výsledný dokument.
XML	Obecný značkovací jazyk.

1 Úvod

Cílem této bakalářské práce je vytvořit webovou aplikaci s využitím relační databáze sloužící jako internetová půjčovna hudebních a filmových médií na CD a DVD nosičích. Aplikace by měla sloužit všem lidem, kteří si chtějí pohodlně z domova půjčit filmy či hudební alba. Měla by umožnit snadné a intuitivní prohlížení a vyhledávání informací o médiích, které je možné si zapůjčit a stejně tak dodatečných informací o hudebních skupinách, hercích a režisérech. Dále by měla umožnit jednoduchou registraci uživatelů do systému se správou kontaktních údajů a přehledem o probíhajících i vyřízených výpůjčkách a upomínkách, a pohodlnou komunikaci s obchodníky pomocí aktualit a dotazního formuláře.

V první části aplikace bude proveden rozbor databázových systémů Oracle a MySql s porovnáním některých jejich funkcí a možností.

V druhé části bude provedena analýza projektu. Na základě této analýzy bude navržena databáze splňující požadavky aplikace. Budou popsány jednotlivé databázové tabulky, jiné databázové objekty a jejich implementace v databázové systému Oracle.

Ve třetí části budou uvedené postupy použité pro vývoj aplikace. Bude vysvětlena práce s použitými technologiemi a popsány důležité funkce aplikace sloužící jako elektronická DVD půjčovna.

V poslední části budou informace o konečném stavu a zhodnocena výsledná aplikace.

2 Analýza databázových systémů Oracle a MySQL

Před začátkem tvorby jakékoliv aplikace používající databázový systém, je třeba se nejdříve zamyslet, který databázový systém použít. Možností je mnoho, záleží na konkrétních potřebách. Zde jsou popsány 2 systémy, mezi kterými je rozhodováno o použití na tento projekt.

2.1. Databázový systém MySQL

„MySQL je databázový systém, vytvořený švédskou firmou MySQL AB, nyní vlastněný společností Sun Microsystems, dceřinnou společností Oracle Corporation. Je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. Komunikace s ním probíhá pomocí jazyka SQL. Podobně jako u ostatních SQL databázových systémů se jedná o dialekt tohoto jazyka s některými rozšířeními. Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace Linux, MySQL, PHP a Apache jako základní software webového serveru. MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly uživatelům produktu již poněkud scházet. Databázový systém MySQL v dnešní době používá většina malých a středních firem.“[1]

2.2. Databázový systém Oracle

Oracle je databázový systém vytvořený firmou Oracle Corporation. Tento databázový systém je považován za světovou špičku ve svém oboru s téměř 49% podílem na trhu. (duben 2010). Je dostupný v několika edicích pod komerční licencí a v edici express edition (XE), která je zcela zdarma a plně využívá databázové jádro. Nicméně má omezení ohledně hardware a OS serveru na kterém běží a dalších doplňků a podpor. Komunikace s ním probíhá stejně jako v případě MySQL pomocí jazyka SQL. Lze jej implementovat na Windows x86 a Linux x86 a v případě komerční licence i na UNIX a 64bitové verze těchto systémů. Lze jej stejně jako MySQL snadno aplikovat na webový server ve spojení s PHP a apache. Tento databázový systém se proslavil především v oblasti zpracování velkých objemů dat, podpory prostředí s vysokými počty transakcí, vysokými požadavky na zabezpečení dat a mechanismy řízení současného přístupu, které umožňují efektivně zvládat vysoké počty transakcí s minimem vzájemného blokování. „Základem je zamykání záznamů na úrovni řádků

[1] Wikipedie, otevřená encyklopedie[online]. [cit. 2010-04-23]. 2001-2010. Dostupný z WWW: http://cs.wikipedia.org/wiki/Mysql/, kap. MySQL

a tzv. Multi-Version Read Consistency, mechanismus, který zajišťuje, že nedochází k vzájemnému blokování zápisových a čtecích operací.“[2]

2.3. Tvorba tabulek

Tabulka je základní databázový objekt, ve kterém jsou uchovávána data. Obsahuje atributy na pozici sloupců a jejich hodnoty na pozici řádků. Ke každému atributu je přiřazen datový typ specifikující jakých hodnot může nabývat a může k němu být přidán modifikátor, jež zajišťuje určitá omezení. Spojením primárního a cizího klíče je definováno spojení nadřazené a podřazené tabulky, kde například nemůže být v nadřazené tabulce vymazán řádek, na nějž se odkazuje cizím klíčem řádek v podřízené tabulce. Toto spojení se může uskutečnit i v jediné tabulce, kde pak jednotlivé záznamy tvoří hierarchické uspořádání.

2.3.1 Oracle

Příklad příkazu pro tvorbu tabulky.

```
CREATE TABLE podrazena_tabulka(
Cislo_karty NUMBER(6) NOT NULL PRIMARY KEY,
Datum DATE NOT NULL,
Nazev VARCHAR(50),
FOREIGN KEY (atribut) REFERENCES nadrazena_tabulka(primarni_klic_tabulky)
);
```

Integritní omezení: NOT NULL – sloupec nesmí obsahovat hodnotu NULL (nic), PRIMARY KEY – primární klíč jednoznačně identifikující řádek tabulky, UNIQUE KEY – sloupec (skupina sloupců) musí být pro všechny řádky v tabulce jedinečný FOREIGN KEY – definuje cizí klíč k jiné tabulce, DEFAULT – obsahuje hodnotu, která bude atributu přiřazena v případě, že nebude žádná vložena, CHECK – uživatelem definované omezení (např. číslo musí být vetší než X, menší než X, v rozsahu X až Y)

2.3.2 MySQL (ver. 5.0.x)

Příklad příkazu pro tvorbu tabulky.

```
CREATE TABLE IF NOT EXIST tabulka (
Cislo INTEGER AUTO_INCREMENT,
Jmeno VARCHAR(50)
)ENGINE InnoDB;
```

[2] http://www.linuxexpres.cz/business/oracledatabase-express-edition, kap. Základní vlastnosti databázo-vého jádra

Na první pohled je zde pár rozdílů. První klauzule, jež se v Oracle nenachází je IF (NOT) EXIST, která v tomto případě zaručuje, že tabulka bude vytvořena pouze pokud ještě neexistuje. Tuto klauzuli lze použít i na tvorbu nové databáze a nebo třeba na mazání tabulky: DROP TABLE IF EXIST.

Další novinkou je zde modifikátor AUTO_INCREMENT, jež zaručí, že se bude hodnota atributu automaticky inkrementovat s každým novým záznamem. Tohoto se výhodně využívá v případě, že je daný atribut primární klíč. Číslo které je modifikátorem AUTO_INCREMENT zvětšováno se dá kdykoliv libovolně nastavit příkazem:

```
ALTER TABLE tabulka AUTO_INCREMENT=50
```

V Oracle tento modifikátor chybí, nicméně jsou zde jiné mechanismy, kterými lze dosáhnout stejného výsledku s možnostmi detailního nastavení. Tyto mechanismy budou popsány v další kapitole.

V MySQL je drobně oslaben i modifikátor DEFAULT, do kteréhož nejde vložit funkce, ale jen konstanta a chybí možnost uživatelsky nastavitelného doménového nastavení CHECK, který například specifikuje rozsah hodnot jednotlivých sloupců. Tento mechanismus, lze obejít pomocí triggerů.

Další zajímavostí je klauzule ENGINE, jež definuje formát úložiště dat v systému MySQL. Těchto formátů je několik, nejdůležitější je zmínit MyISAM, jež je výchozí formát pro každou tabulku pokud její tvůrce nenastaví jinak, a formát InnoDB
Hlavní rozdíly mezi těmito formáty jsou podpora referenční integrity pomocí cizích klíčů a možnost spouštět příkazy jako transakce v tabulce typu InnoDB. Oproti tomu je MyISAM rychlejší, stabilnější, umožňuje vytvářet fulltextový index a případná licence bez podpory InnoDB je znatelně levnější. Další rozdíl je ve způsobu zamykání, kde InnoDB zamyká na úrovni řádků a MyISAM na úrovni celých tabulek.

2.4. Práce se stromovou strukturou dat

V této části jsem čerpal z této literatury: P*okročilý SELECT v SŘBD Oracle* [12]

Stromové datové struktury představují pro relační databáze poměrně těžko stravitelné sousto. Problémy vyplývají ze samotné podstaty relačního databázového modelu, který není pro ukládání tohoto typu dat příliš vhodný. Hierarchicky uspořádaná data je tedy potřeba nejdříve přepsat do plochých relačních tabulek. Stromová struktura takovéto tabulky je naznačena spojením primárních a cizích klíčů v rámci jedné tabulky. Takové tabulce se říká rekurzivní (spojená sama se sebou).

Tabulka 1 – Rekurzivní tabulka

Id_osoby	Jmeno	Prijmení	Id_nadrizeneho	Cislo_karty
1	Vlastimil	Dudacek	null	1
2	Karel	Vavricka	1	4
3	Stanislav	Herout	1	5
4	Maxmilian	Rohlik	1	6

5	Ondrej	Pesicka	2	7
6	Pavel	Bokr	2	8
7	Antonin	Dura	3	null

V této tabulce je cizí klíč *Id_nadrizeného* spojen s primárním klíčem *Id_osoby*. V databázovém systému MySQL a každém jiném lze s touto tabulkou pracovat pomocí běžných prostředků jako jsou jednoduché dotazy, rekurzivní dotazy, spojení tabulky samu se sebou apod. případně lze doplnit do tabulky další údaje pro zjednodušení práce. Nicméně v závislosti na složitosti tabulky a konkrétním dotazu, který chceme provést může být získání výsledků velmi obtížné.

Jednoduchý dotaz, který má vrátit jména všech lidí se jmény jejich přímých nadřízených pomocí rekurzivního spojení tabulky by mohl vypadat takto:

```
SELECT  a.prijmeni, a.jmeno, b.prijmeni, b.jmeno
  FROM  osoby a, osoby b
 WHERE  a.id_nad = b.id_osoby;
```

2.4.1 Hierarchické dotazy v Oracle

Databázový systém Oracle má pro práci s rekurzivními tabulkami specielní konstrukci. Slouží k tomu klauzule **CONNECT BY PRIOR**, kde **PRIOR** označuje stranu rodiče. Výše zmíněný dotaz, který by navíc vypsal všechny nadřízené nikoliv jen do určité hloubky, by s použitím této konstrukce vypadal následovně.

```
SELECT  LEVEL, jmeno, prijmeni
   FROM  osoby
CONNECT  BY id_osoby = PRIOR id_nad;
```

LEVEL je pseudosloupec, jehož hodnota nám říká v jaké úrovni se nacházíme. Pouhým přesunutím slova PRIOR na druhou stranu podmínky získáme ke jménům zaměstnanců jména jejich podřízených

```
SELECT  LEVEL, jmeno, prijmeni
   FROM  osoby
CONNECT  BY PRIOR id_osoby = id_nad;
```

Výše uvedené dotazy prochází strukturu rekurzivně pro každý záznam od „kořene stromu“. Můžeme však specifikovat, odkud se má začít hierarchická struktura prohledávat použitím klauzule **START WITH**. Např. chceme zobrazit hierarchii podřízených od konkrétního zaměstnance.

```
SELECT  LPAD(' ', (LEVEL-1)*8)||jmeno||' '||prijmeni AS osoba
   FROM  osoby
```

```
CONNECT  BY PRIOR id_osoby = id_nad
  START  WITH jmeno = 'Karel' AND prijmeni = 'Vavricka';
```

2.5. Řešení modifikátoru AUTO_INCREMENT v Oracle

Jak již bylo výše zmíněno, často se nám stane, že máme v tabulce primární klíč nějaké číslo. Toto číslo pak musí být v rámci celé tabulky unikátní čehož nejlépe dosáhneme, pokud bude toto číslo inkrementováno s každým přidaným řádkem tabulky. V databázovém systému MySQL je tohoto velmi jednoduše a rychle dosáhnuto přidáním modifikátoru AUTO_INCREMENT k atributu, který má být primárním klíčem. Ten zařídí, že po přidání nového řádku bude hodnota tohoto atributu automaticky vygenerována. V databázovém systému Oracle se o unikátní hodnotu stará objekt sekvence a pro plnou automatizaci generování unikátní hodnoty sekvencí lze použít trigger.

2.5.1 Sekvence

Sekvence je databázový objekt, který generuje automaticky se zvyšující nebo snižující posloupnost čísel. Nejčastěji se používá pro generování hodnot číselných primárních klíčů. Pro každou takovou tabulku je pak vytvořena vlastní sekvence. Unikátní číslo se pak generuje příkazem NazevSekvence.NEXTVAL a zároveň si pak každá sekvence pamatuje poslední vygenerovanou hodnotu, kterou lze získat příkazem NazevSekvence.CURRVAL. při tvorbě sekvence ji lze nadefinovat přesně podle aktuálních potřeb. Lze nastavit minimální a maximální generovanou hodnotu, krok se kterým se bude hodnota zvyšovat či snižovat, číslo kterým bude sekvence začínat generovat hodnoty, zda se má generovaní hodnot cyklicky opakovat, kolik má mít sekvence v zásobě hodnot a řád podle kterého se hodnoty generují. Příklad vytvoření sekvence:

```
CREATE SEQUENCE NazevSekvence NOCYCLE NOORDER CACHE 20 NOMAXVALUE
MINVALUE 1 INCREMENT BY 1 START WITH 1
```

Jako doplněk se ke každé sekvenci tvoří trigger, který před každým přidání řádku do tabulky ke které se sekvence používá vygeneruje novou hodnotu, kterou načte do příslušného atributu. Jako příklad je zde tabulka *a_aktuality*, kde je primární klíč automaticky generované pořadové číslo sekvencí *saktuality*:

```
create or replace TRIGGER "z_aktuality"
before insert on a_aktuality
for each row
begin
 select saktuality.nextval INTO :new.id_aktuality from dual;
end;
```

Tento systém generování automatických posloupností plně nahrazuje modifikátor AUTO_INCREMENT, jež je využíván v systému MySQL a přesto, že je tato metoda o něco více pracná, je zde možnost si sekvenci nastavit do posledního detailu, aby vyhovovala svému účelu a generování lze provádět i manuálně.

2.6. Omezení počtu záznamů – Limity

Velmi často se můžeme setkat s případy, že je při výpisu dat z databáze příliš mnoho položek (řádků) a je třeba jednotlivé položky zařadit do stránek. K tomu je třeba nastavit SELECT tak, aby vypsal jen požadovaný počet záznamů a to až od určitého čísla vypisované položky.

2.6.1 Limity v MySQL

Databázový systém MySQL řeší omezení výpisu dat z databáze selectem pomocí klauzule LIMIT, za kterou lze dosadit jeden nebo dva parametry. V případě, že bude uveden pouze jeden parametr, znamená to maximální počet prvků který má uvedený select vrátit od prvního vypisovaného prvku. S tímto je třeba použít také klausuli ORDER BY, neboť select vrací výsledky většinou v pořadí v jakém byly záznamy do databáze vloženy.

```
SELECT * FROM a_aktuality LIMIT 10;
```

Tento select vypíše prvních 10 položek z tabulky *a_aktuality*. V případě, že za klausuli limit vložíme dva parametry, bude první parametr udávat od které položky začít vypisovat a druhý parametr bude udávat počet prvků, které se mají vypsat.

```
SELECT * FROM a_aktuality LIMIT 50,10;
```

Tento select vypíše položky 51 až 60.

2.6.2 Limity v Oracle

Databázový systém Oracle nemá klausuli LIMIT. Omezení výpisu z databáze selectem je zde řešeno vnořeným selectem a přidanou podmínkou. Existuje zde pseudosloupec **ROWNUM** ve kterém je pořadové číslo vypisované položky. Výpis konkrétních položek je pak možný přidáním podmínky, které budou vyhovovat pouze položky s konkrétním pořadovým číslem.

```
SELECT * from (SELECT rownum r_ ,id_aktuality,nadpis,datum,nick,obsah from a_aktuality) WHERE
r_ BETWEEN (50) AND (60)
```

Tento select vypíše položky 51 až 60. Toto řešení je složitější než v případě MySQL nicméně je stejně účinné a u každého řádku je dané jeho číslo, což může být dále využitelné.

2.7. Databáze a PHP

Většina aplikací běžících na internetu bývá programována pomocí jazyka PHP. V případě, že tyto aplikace využívají databázový server k úschově dat, je třeba, aby se s ní dalo pomocí tohoto jazyku komunikovat.

2.7.1 MySQL

Do verze PHP5 mělo MySQL v PHP jako jediný databázový systém interní podporu, tzn. že používat ve webové aplikaci MySQL databázi šlo ihned. Od PHP5 již tato podpora není kvůli GPL licenci pod kterou je MySQL distribuován.

Pro zprovoznění MySQL podpory na webovém serveru je tedy potřeba na správné místo umístit knihovnu **libmySQL.dll,** v konfiguračním souboru **php.ini** přidat (odkomentovat) **extension=php_mysql.dll** a samozřejmě mít nainstalovaný a zprovozněný MySQL databázový server. V tuto chvíli je v PHP aplikaci možnost používat funkcí, které přímo pracují s databázovým systémem MySQL.

První funkcí, která je potřeba použít v každé webové aplikaci je vlastní připojení k fyzické databázi.

```
@$con=mysql_pconnect($db,$user,$passw) or die(„Server nepřipojen“);
```

@ - tichý režim – nevypíší se chyby
$con – proměnná identifikující konkrétní databázi
mysql_pconnect – funkce, která naváže spojení s DB serverem (Zadává se umístění serveru, DB uživatel a jeho heslo)
or die – v případě neúspěchu se vypíše daný řetězec

```
@$db=mysql_select_db($jmeno,$con);
```

Otevře konkrétní databázi. Lze obalit podmínkou, zda byl pokus úspěšný.

```
$res=mysql_query($dotaz,$con);
```

Základní funkce pro komunikaci s databází. Provede SQL dotaz zaznamenaný v proměnné *$dotaz* v rámci připojení s číslem *$con*.

```
$ret1=mysql_real_escape_string($ret,$con),
```

Tato funkce ošetří řetězec *$ret* a tento vrátí do proměnné *$ret1*. Připojení není nutné pokud bylo provedeno někde výše. Slouží k obraně před SQL injection.

```
$res=mysql_query($dotaz,$con);
while($data=mysql_fetch_array($res)){}
```

Základní výpis všech výsledků z databáze. Aktuální řádek výpisu je v cyklu načten do proměnné PHP *$data*. Tato proměnná je v tuto chvíli pole, jehož položky jsou názvy sloupců tabulky, ze které je dotaz prováděn.

2.7.2. Oracle

PHP je samozřejmě schopné komunikovat i s databázových systémem Oracle, nicméně nastavení je oproti MySQL nepatrně složitější. Možností je více, zde bude nastíněna práce pomocí knihovny OCI8.

Pro zprovoznění podpory Oracle na webovém serveru je třeba nakopírovat na příslušné místo soubor **oci.dll** a stáhnout **Instantclient** ze stránek Oracle, jehož cestu je třeba zadat do systémových cest OS. Dále pak v příslušných konfiguračních souborech **PHP.ini** povolit (odkomentovat) řádek **extension=php_oci8.dll; Use with Oracle 10gR2 Instant Client** a samozřejmě mít nainstalovaný databázový server. V tuto chvíli je v PHP aplikaci možnost používat sadu funkcí, které přímo pracují s databázovým systémem Oracle.

Stejně jako v MySQL i zde je první a nejdůležitější funkce vytvoření fyzického spojení s databází.

```
@$con = oci_conect($user,$password,$conect) or die („Server nepřipojen“);
```

Jednotlivé části příkazu mají stejný účel jako v případě MySQL, kde jsou rozepsány.

```
$res=oci_parse($con,$dotaz);
oci_execute($res);
```

Dvě základní funkce pro komunikací s databází, první řádek připraví dotaz v proměnné *$dotaz* v rámci připojení s číslem *$con* a druhý řádek daný dotaz vykoná a automaticky comitne (uloží změny v DB).

```
$res=oci_parse($con, „insert into a_tabulka (cislo,jmeno,prijmení)values(1,:ret1,:ret2)“);
oci_bind_by_name($res,“:ret1“,$ret1);
oci_bind_by_name($res,“:ret2“,$ret2);
oci_execute($res);
```

Takto vypadá vkládání dat do tabulky *a_tabulka* s ošetřením proti SQL injection. Do DB jsou hodnoty PHP proměnných nejdříve mapovány příkazem *oci_bind_by_name*. Pokud máme sadu funkci SQLlite lze provést přímo ošetření řetězce použitím funkce

sqlite_escape_string(), která převede nebezpečné znaky na takzvané escape sekvence tzn. přidá před ně zpětné lomítko.

```
$res=oci-parse($con,$dotaz);
while($data=oci_fetch_array($res)){}
```

Podobně jako v MySQL, výpis položek z DB v cyklu. Zde je ale třeba si dát pozor, neboť název sloupce v proměnné *$data* je nutno psát vždy velkými písmeny.

2.8. Závěr

Cílem tohoto rozboru bylo porovnání databázových systému Oracle a MySQL a vhodnost těchto systémů pro použití v běžných webových aplikacích. Práce v obou systémech je podobná, nicméně některé funkcionality se zde řeší jinak. Propojení a podpora obou systémů v PHP je vyhovující. U většiny malých a středních projektů lze bez problémů použít oba systémy zdarma bez obav, že by se v jednom z nich nedal vyřešit jakýkoliv běžný problém, zatímco v druhém systému ano.

Pro potřeby internetové DVD půjčovny lze bez obav použít oba databázové systémy. Nakonec byl vybrán databázový systém Oracle hlavně proto, že se na Univerzitě Pardubice tento systém dva semestry vyučuje a moje zkušenosti s ním jsou větší, než v případě MySQL.

3 Analýza projektu

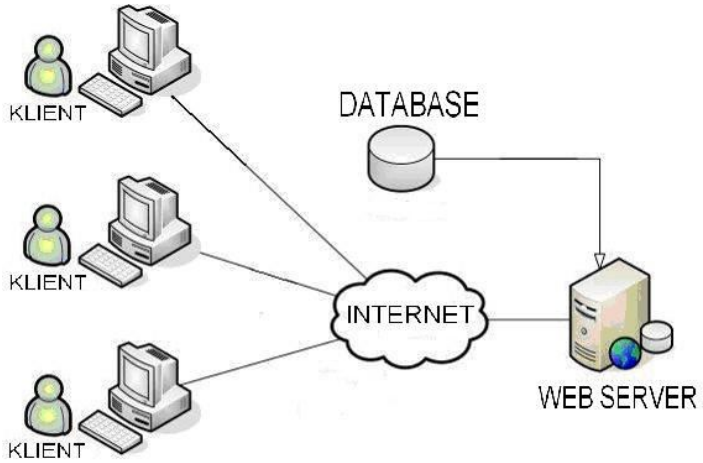
Cílem je vytvořit jednoduchou elektronickou půjčovnu hudebních a filmových médií. Projekt se skládá z webové aplikace a databázového serveru, do kterého aplikace přistupuje. Aplikace umožňuje registraci uživatelů, efektivní vyhledávání dat, vytváření výpůjček médií, rozsáhlou administraci, a obsahuje oddělené funkce a možnosti na základě 4 typů uživatelů: administrátor, obchodník, registrovaný uživatel a anonymní uživatel. Aplikace bude vyžadovat dynamicky se měnící obsah, a proto bude programovaná v jazyce PHP. Pro uložení, snadnou a bezpečnou manipulaci s daty bude použit databázový systém Oracle 10g, neboť s ním mám největší zkušenosti.

3.1. Architektura aplikace

Aplikace využívá síťovou architekturu klient-server. Jednotliví uživatelé (klienti) komunikují přes internetovou síť s aplikací, která je umístěna na webovém serveru, zpracovává požadavky jednotlivých uživatelů, komunikuje s databázovým serverem a zprostředkovává uživatelům požadovaná data formou webové prezentace.

Pro vývoj aplikace byl použit internetový klient Mozzila Firefox verze 3.6.3, multifunkční textový editor PSPad verze 4.5.3, databázový server Oracle 10g a webserverový program WAMP zahrnující webový server Apache verze 2.2.11 a PHP server verze 5.3.0. Výše uvedené programy jsou multiplatformní a pro jejich užívání je zvolen Microsoft Windows XP x86 CZ + Sp2. Do aplikace jsou dále přidány doplňky

jako WYSIWYG editor Tiny_MCE a reCAPTCHA klienta pro test odlišení člověka od počítače.



Obrázek 1- Architektura klient – server

3.2. Uživatelé v aplikaci

Aplikace je postavena na principu uživatelských práv. Jsou rozlišováni 4 typy uživatelů, kterým je po přihlášení zpřístupněna konkrétní část aplikace a mají specifické možnosti jejího využívání.

Anonymní uživatel – může prohlížet a vyhledávat všechny média v databázi, prohlížet aktuality a další informace, a má možnost se zaregistrovat prostřednictvím registračního formuláře.

Registrovaný uživatel – má stejná práva jako anonymní uživatel, navíc však vidí počet kusů jednotlivých médií na skladě, má k dispozici nákupní košík do kterého lze přidávat libovolné zboží z nabídky, může provést objednávku, upravovat svoje informace z registrace, sledovat stav svých výpůjček a upomínek, a prostřednictvím kontaktního formuláře poslat dotaz obchodníkům

Obchodník – může přidávat a mazat aktuality a změnit jejich obsah či nadpis pokud se jedná o aktualitu jím napsanou. Dále se stará o přidávání, mazání a editaci filmů, hudebních skupin a jejich alb, režisérů a herců. Pak má k dispozici také vyřizování objednávek, upomínek a dotazů od registrovaných uživatelů a může editovat informace a kontakty ohledně půjčovny.

Administrátor – má stejná práva jako obchodník a navíc může editovat všechny aktuality, mazat uživatelské účty a měnit jejich práva.

3.3. Rich picture diagram

Rich picture diagram je přiložen v příloze B.

3.4. UML usecase diagram

UML usecase diagram je přiložen v příloze C.

3.5. UML activity digram

UML activity diagram je přiložen v příloze D.

4 Návrh databáze

Po návrhu projektu, analýze rozsahu a obsahu aplikace je vhodné přejít k návrhu kvalitní databáze. Databáze bude obsahovat velké množství informací o filmech, osobních údajích uživatelů a dalších, a často v ní bude vyhledáváno. Proto je důležité, aby byly tabulky správně dekomponované, neobsahovaly redundantní a jiné zbytečné data, byly bezpečné a aby byla správně zajištěna referenční integrita.

4.1. Konceptuální datový model

Pro vytvoření správné databáze je potřeba navrhnout dobrý konceptuální datový model, který se zabývá entitami, jejich vzájemnými vztahy a jejich daty a atributy.

Entita – tabulka odpovídající reálnému objektu se správně zvolenými atributy uchovávající potřebné informace o objektu. V tomto smyslu by mohla být entita tabulka s filmy.

Atribut – vlastnost nějaké entity o které potřebujeme uschovat data. Atribut filmu může být jeho název.

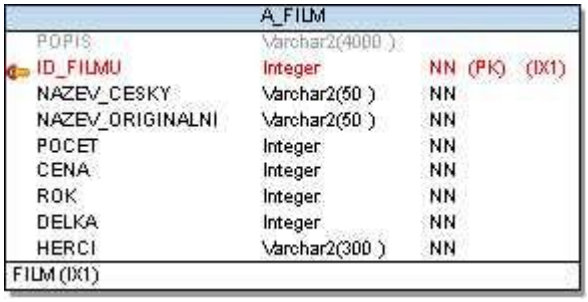
Kardinalita – vyjádření vztahu mezi entitami určující kolik záznamů jedné entity je ve vztahu k záznamům entity druhé. Existují tři druhy kardinality: 1:1, 1:N a M:N. 1:1 znamená, že jeden řádek v první entitě odpovídá jednomu řádku v druhé entitě. 1:N znamená, že jeden řádek v první entitě může odpovídat více řádkům v druhé entitě, například jeden režisér může být přiřazen více filmům. Poslední případ může nastat u entity herec, kde více herců může hrát ve více filmech a řeší se vytvořením pomocné tabulky, která má k oběma původním vztah 1:N a obsahuje pouze klíče obou původních tabulek pro jejich logické propojení.

Pro vyjádření konceptuálního modelu je nejlepší ER model pro jeho jednoduchost, přehlednost a nezávislost na zvoleném databázovém systému. Nyní je třeba převést ER diagram do relačního modelu skrze normální formy tzv. normalizací. Normalizace je postup při kterém se dekomponují jednotlivé entity tak, aby splňovaly

jisté pravidla. Pro běžnou a efektivní databázi bude stačit převést entity do třetí normální formy, nicméně normálních norem je ještě více.

4.2. Nultá normální forma

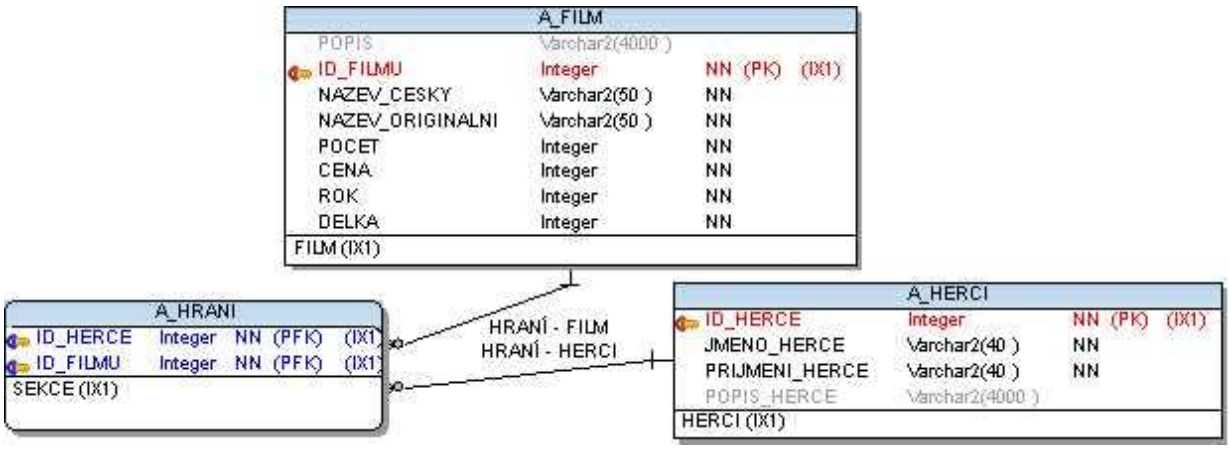
„Tabulka je v nulté normální formě právě tehdy, když existuje alespoň jedno pole, které obsahuje více než jednu hodnotu.“[3] Tabulka *a_film* obsahuje neatomické atributy, například seznam všech herců.



Obrázek 2 - Tabulka v nulté normální formě

4.3. První normální forma

Tabulka je v první normální formě právě tehdy, když je v nulté normální formě a obsahuje pouze dále nedělitelné (atomické) atributy. U tabulky *a_film* se předpokládá, že každý film má více herců. Atribut *herci* je tudíž neatomický a velmi obtížně by se v něm hledalo a navíc je možné, že každý herec hraje ve více než jednom filmu, což způsobuje redundanci dat. Proto byla vytvořena tabulka *a_herci* a kvůli vazbě M:N ještě umělá tabulka *a_hrani,* jež je k oběma tabulkám vázána identifikující vazbou, což znamená, že cizí klíče v obou tabulkách vytváří složený primární klíč v tabulce *a_hrani*.

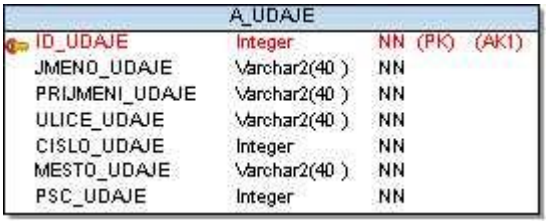


Obrázek 3 - Tabulka v první normální formě

[3] http://www.penguin.cz/noviny/chip/sql/SQL2.pdf, kap. Nultá a první normální forma

4.4. Druhá normální forma

„Tabulka je v druhé normální formě, jestliže je v první normální formě, zároveň existuje klíč a současně všechna neklíčová pole jsou funkcí celého klíče, a nikoliv jen jeho části.“[4] Jelikož tabulka *a_uzivatel* obsahuje pouze jednoduchý primární klíč a je nyní v první normální formě, automaticky splňuje podmínky pro druhou normální formu.

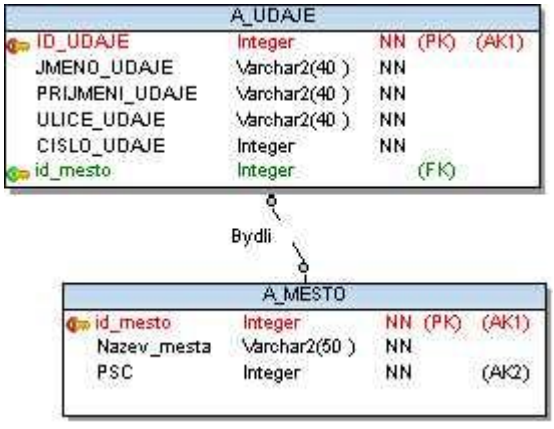


Obrázek 4 - Tabulka v druhé normální formě

4.5. Třetí normální forma

„Tabulka je ve třetí normální formě, jestliže je ve druhé normální formě a zároveň neexistují závislosti neklíčových sloupců tabulky.“[5] U tabulky *a_udaje* atribut *psc_udaje* nezávisí na primárním klíči *id_udaje*, nýbrž na názvu města a nesplňuje tudíž třetí normální formu. Pro město je tedy třeba vytvořit novou tabulku, kde bude jeho název a PSČ a tu poté spojit s tabulkou *a_udaje*.

Tato dekompozice však nebude nutná pro správnou funkci databáze pro aplikaci DVD půjčovny.



Obrázek 5 - Tabulka v třetí normální formě

[4] http://www.penguin.cz/noviny/chip/sql/SQL3.pdf, kap. Druhá a třetí normální forma

[5] http://www.penguin.cz/noviny/chip/sql/SQL3.pdf, kap. Druhá a třetí normální forma

4.6. Fyzický datový model

Výsledný ER diagram je přiložen v příloze E.

5 Databáze

Databáze pro elektronickou půjčovnu obsahuje 23 tabulek s textovými daty. Přestože je třeba uschovávat i obrázky, byla zvolena metoda, při které je v databázi uschována pouze cesta k obrázku, ale obrázek samotný je uložen mimo ní. Mimo tabulek je třeba použít i jiné databázové objekty: indexy, procedury, funkce, triggery a sekvence.

Vývoj databáze probíhal vytvořením fyzického modelu v programu Toad Data Modeler, v němž je možné jednoduše graficky navrhnout jednotlivé tabulky se všemi atributy a dané tabulky pak pospojovat patřičnými vazbami. Vytvořený model byl poté verifikován pro odstranění chyb při návrhu, aby bylo možné danou databázi vytvořit na databázovém serveru. Dále byl programem vygenerován soubor s SQL skriptem sloužícím pro přenos do databázového systému. Pro práci s fyzickou databází byl použit program SQL Developer. V něm byl spuštěn vytvořený skript, který vytvořil samotnou databázi a dále sloužil pro tvorbu ostatních databázových objektů, drobným úpravám a kontrole dat během vývoje celé aplikace.

Vlastní databáze byla uložena na Oracle 10g Express Edition, což je volně šířitelná verze databáze od firmy Oracle, jehož produkty jsou ve světě velmi známé pro svojí kvalitu, spolehlivost a výkon. Protože je tato verze zdarma ke stažení, jsou zde určitá omezení. Lze nainstalovat pouze na 32bitový OS Windows a Linux, využívá pouze jedno jádro vícejádrového procesoru, 1 GB operační paměti a vlastní data v ní uložené nemůžou přesáhnou 4 GB. Pro účely elektronické DVD půjčovny však tato verze plně dostačuje.

5.1. Tabulky v databázi

5.1.1 Tabulka A_AKTUALITY

Obrázek 6 - Tabulka A_aktuality

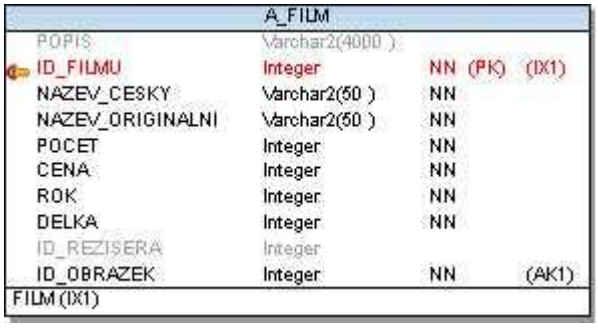
Tabulka *a_aktulity* obsahuje informace od obchodníků a administrátorů pro registrované i anonymní uživatele. Atribut *nick* je zde cizím klíčem na tabulku *a_uzivatel*. Tato tabulka jako mnoho dalších obsahuje primární klíč umělý atribut ID_aktuality, jež byl do tabulky přidám pouze k účelu jednoznačného identifikování řádku tabulky. Pro jednoduchou manipulaci s primárním klíčem je vytvořena u každé takovéto tabulky sekvence, která zajišťuje, že každá nová hodnota v atributu *ID_Aktulity* bude skutečně unikátní .

```
CREATE SEQUENCE SAKTUALITY MINVALUE 1 INCREMENT BY 1 NOCACHE
NOCYCLE;
```

Pro kompletní automatizaci umělých cizích klíčů je u každé sekvence trigger, který je aktivován před přidáním nového řádku tabulky. Tento trigger sám zajistí automatické vygenerování nového primárního klíče.

```
create or replace TRIGGER "z_aktuality"
before insert on a_aktuality
for each row
begin
 select saktuality.nextval INTO :new.id_aktuality from dual;
end;
```

5.1.2. Tabulka A_FILM



Obrázek 7 - Tabulka A_FILM

Tabulka *a_film* obsahuje informace o filmových médiích, cenu, počet kusů na skladě a cizí klíče na tabulky *a_reziseri* a *a_obrazky*. Primární klíč je zde umělý atribut *id_filmu*. Při odebírání z této tabulky jsou aktivovány 2 triggery, které pomáhají udržet referenční integritu mezi řádky této tabulky a řádky podřízených tabulek, jež se na tuto

odkazují cizím klíčem. Jeden se aktivuje před a druhý po odebrání z nadřazené tabulky a jejich úkol je mazat patřičné a již bezcenné řádky v podřízených tabulkách.

```
create or replace TRIGGER "PREDODEBERFILM"
before delete on a_film
for each row
begin
 delete from a_hrani where id_filmu = :OLD.id_filmu;
 delete from a_zobrazeni where id_filmu = :OLD.id_filmu;
 delete from a_prislusnost where id_film = :OLD.id_filmu;
 delete from a_obrazky where id_filmu = :OLD.id_filmu;
 delete from a_vypujcka where id_vypujcka IN (select id_vypujcka from a_kosik where id_filmu=:OLD.id_filmu);
end;
```

```
create or replace TRIGGER "POODEBERFILM"
after delete on a_film
for each row
begin
 delete from a_obrazky where id_obrazek=:OLD.id_obrazek;
end;
```

5.1.3. Tabulka A_HERCI

Obrázek 8 - Tabulka A_HERCI

Tabulka *a_herci* obsahuje informace o hercích v databázi. Primární klíč je zde umělý atribut *id_herce*. Při odebírání z této tabulky jsou aktivovány 2 triggery pro udržení referenční integrity v podřízené a nadřízené tabulce.

```
create or replace TRIGGER "PREDODEBERHERCE"
before delete on a_herci
for each row
begin
 delete from a_hrani where id_herce=:OLD.id_herce;
end;
```

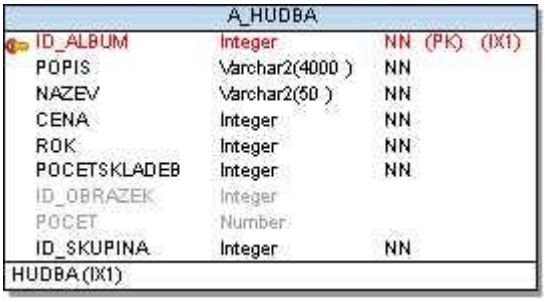
```
create or replace TRIGGER "POODEBERHERCE"
after delete on a_herci
for each row
begin
 delete from a_obrazky where id_obrazek = :OLD.id_obrazek;
end;
```

5.1.4. Tabulka A_HRANÍ

Obrázek 9 - Tabulka A_HRANÍ

Tabulka *a_hraní* je uměle vytvořená tabulka reprezentující vazbu M:N mezi tabulkami *a_herci* a *a_film,* neboť v jednom filmu může hrát více herců a zároveň jeden herec může hrát ve více filmech. Oba atributy v této tabulce jsou cizí klíče na výše zmíněné tabulky a kombinace těchto dvou atributů tvoří dohromady složený primární klíč.

5.1.5. Tabulka A_HUDBA



Obrázek 10 - Tabulka A_HUDBA

Tabulka a_hudba obsahuje informace o jednotlivých hudebních kompilacích, počet kusů na skladě a cizí klíče na titulní obrázky a hudební skupiny. Primární klíč je zde umělý atribut *id_album*. Při odebírání z této tabulky jsou aktivovány 2 triggery pro udržení referenční integrity v podřízených tabulkách.

```
create or replace TRIGGER "PREDODEBERALBUM"
before delete on a_hudba
for each row
begin
 delete from a_zobrazenih where id_alba=:OLD.id_album;
 delete from a_obrazky where id_alba=:OLD.id_album;
end;
```

```
create or replace TRIGGER "POODEBERALBUM"
after delete on a_hudba
for each row
begin
 delete from a_obrazky where id_obrazek = :OLD.id_obrazek;
 delete from a_vypujcka where id_vypujcka IN (select id_vypujcka from a_kosik where id_alba=:OLD.id_album);
end;
```

5.1.6. Tabulka A_INFO

Obrázek 11 - Tabulka A_INFO

Tabulka *a_info* obsahuje pouze jeden sloupec a jeden řádek. Je zde uchovaný editovatelný informační text o půjčovně, jež se uživatelům zobrazí v sekci *kontakty*. Tabulka neobsahuje primární klíč a není propojena s žádnou jinou tabulkou.

5.1.7. Tabulka A_KOSIK

Obrázek 12 - Tabulka A_KOSIK

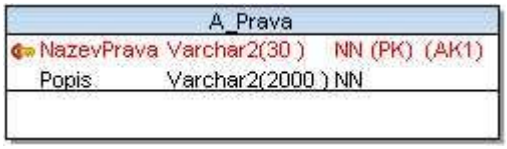
Tabulka *a_kosik* obsahuje uměle vytvořenou vazbu mezi tabulkou *a_vypujcka* a tabulkami *a_film* a *a_hudba*. Platí, že jedna výpůjčka může obsahovat více zboží a každé zboží může být na více výpůjčkách. Přesto zde není složený primární klíč z *id_vypujcka, id_filmu* a *id_alba* nýbrž umělý atribut id_kosik v roli primárního klíče, id_vypujcka jako cizí klíč na tabulku *a_vypujcka* a atributy *id_filmu* a *id_alba* jejichž referenční integritu zajišťují triggery v nadřazených tabulkách, neboť část složeného klíče by byla vždy hodnoty NULL, protože jedna položka v košíku nemůže být zároveň film a zároveň hudební album. Před přidáním položky do košíku je aktivován trigger, jež upraví počet kusů daného zboží na skladě a automaticky počítá cenu vznikající výpůjčky. Tento trigger je umístěn v příloze F.

5.1.8. Tabulka A_OBRAZKY

Obrázek 13 - Tabulka A_OBRAZKY

Tabulka *a_obrazky* obsahuje data o titulních obrázcích ke všem filmům, albům, skupinám, hercům a režisérům. Pro tyto obrázky je tato tabulka nadřízená. Dále zde jsou obrázky do galerie pro filmy a hudební alba, pro které je tato tabulka podřízená. Primární klíč je zde umělý atribut *id_obrazek*.

5.1.9. Tabulka A_PRAVA



Obrázek 14 - Tabulka A_PRAVA

Tabulka *a_prava* obsahuje data o druzích práv jež definují možnosti jednotlivých uživatelů. Primární klíč je zde atribut *NazevPrava*.

5.1.10. Tabulka A_PRISLUSNOST

Obrázek 15 - Tabulka A_PRISLUSNOST

Tabulka *a_prislusnost* je uměle vytvořená tabulka reprezentující vazbu M:N mezi tabulkami *a_staty* a *a_film,* neboť jeden film může vzniknout ve více státech a zároveň v jednom státě může vzniknout více filmů. Oba atributy v této tabulce jsou cizí klíče na výše zmíněné tabulky a kombinace těchto dvou atributů tvoří dohromady složený primární klíč.

5.1.11. Tabulka A_PRISLUSNOSTH

Obrázek 16 - Tabulka A_PRISLUSNOSTH

Tabulka *a_prislusnosth* je uměle vytvořená tabulka reprezentující vazbu M:N mezi tabulkami *a_staty* a *a_hudba,* neboť jedno hudební album může vzniknout ve více státech a zároveň v jednom státě může vzniknout více hudebních alb. Oba atributy v této tabulce jsou cizí klíče na výše zmíněné tabulky a kombinace těchto dvou atributů tvoří dohromady složený primární klíč.

5.1.12. Tabulka A_REZISERI

Obrázek 17 - Tabulka A_REZISERI

Tabulka *a_reziseri* obsahuje informace o režisérech v databázi. Primární klíč je zde umělý atribut *id_rezisera*. Při odebírání z této tabulky je aktivován trigger pro udržení referenční integrity v nadřízené tabulce.

```
create or replace TRIGGER "POODEBERREZISERA"
after delete on a_reziseri
for each row
begin
 delete from a_obrazky where id_obrazek = :OLD.id_obrazek;
end;
```

5.1.13. Tabulka A_SEKCE

Obrázek 18 - Tabulka A_SEKCE

Tabulka *a_sekce* obsahuje data o filmových žánrech podle kterých se dají všechny filmy třídit. Primární klíč je zde umělý atribut *id_sekce*.

5.1.14. Tabulka A_SEKCEH

Obrázek 19 - Tabulka A_SEKCEH

Tabulka *a_sekceh* obsahuje data o hudebních žánrech, podle kterých se dají všechny hudební alba třídit. Primární klíč je zde umělý atribut *id_sekceh*.

5.1.15. Tabulka A_SKUPINY

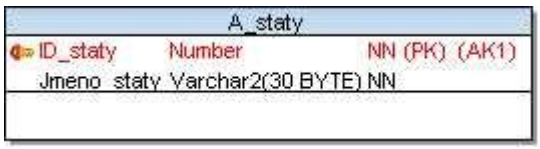
Obrázek 20 - Tabulka A_SKUPINY

Tabulka a_skupiny obsahuje informace o hudebních skupinách. Primární klíč je zde umělý atribut *id_skupina*. Při odebírání z této tabulky jsou aktivovány dva triggery pro udržení integrity v podřízených a nadřízených tabulkách.

```
create or replace TRIGGER "PREDODEBERSKUPINU"
before delete on a_skupiny
for each row
begin
 delete from a_prislusnosth where id_skupina=:OLD.id_skupina;
 delete from a_hudba where id_skupina=:OLD.id_skupina;
end;
```

```
create or replace TRIGGER "POODEBERSKUPINU"
after delete on a_skupiny
for each row
begin
 delete from a_obrazky where id_obrazek=:OLD.id_obrazek;
end;
```

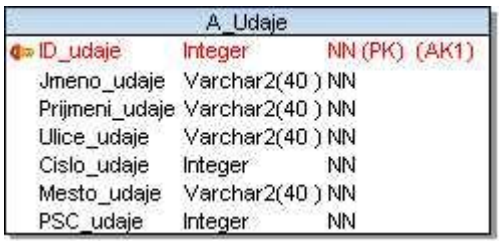
5.1.16. Tabulka A_STATY



Obrázek 21 - Tabulka A_STATY

Tabulka *a_staty* obsahuje seznam států. Primární klíč je zde umělý atribut *id_staty*.

5.1.17. Tabulka A_UDAJE



Obrázek 22 - Tabulka A_UDAJE

Tabulka *a_udaje* obsahuje informace o kontaktních údajích uživatelů pro jejich objednávky. Tabulka je spojena s tabulkou *a_uzivatel* vazbou 1:1. Primárním klíčem je umělý atribut *id_udaje*.

5.1.18. Tabulka A_UPOMINKY

Obrázek 23 - Tabulka A_UPOMINKY

Tabulka *a_upominky* obsahuje informace o výpůjčkách, jež svým trváním přesáhly určitou dobu. Primární klíč je zde umělý atribut *id_vypujcky*.

5.1.19. Tabulka A_UZIVATEL

Obrázek 24 - Tabulka A_UZIVATEL

Tabulka *a_uzivatel* obsahuje data o všech uživatelích. Primárním klíčem je zde atribut *nick*. Jelikož přihlašovací jméno uživatele musí být unikátní, nebylo zde nutné tvořit umělý atribut. Stejně tak musí být unikátní i e-mail pomocí kterého se resetuje heslo. Heslo samotné se v databázi uchovává v podobě md5 hashe.

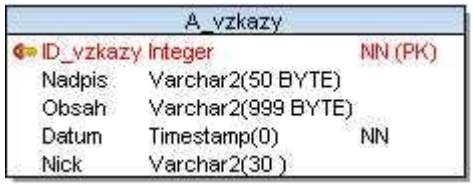
5.1.20. Tabulka A_VYPUJCKA

Obrázek 25 - Tabulka A_VYPUJCKA

Tabulka *a_vypujcka* obsahuje informace o provedených objednávkách. Primární klíč je zde umělý atribut *id_vypujcka*. Samotný obsah výpůjček je v podřízené tabulce *a_kosik* a podle atributu *cas* se v podřízené tabulce *a_upominky* automaticky generují upomínky k jednotlivých výpůjčkám. Před odebírání z této tabulky je aktivován trigger který pomáhá udržovat referenční integritu smazáním patřičných řádku v podřízených tabulkách.

```
create or replace TRIGGER "PREDODEBERVYPUJCKU"
before delete on a_vypujcka
for each row
begin
 delete from a_upominky where id_vypujcky=:OLD.id_vypujcka;
 delete from a_kosik where id_vypujcka=:OLD.id_vypujcka;
end;
```

5.1.21. Tabulka A_VZKAZY



Obrázek 26 - Tabulka A_VZKAZY

V tabulce *a_vzkazy* jsou uchovávány informace o dotazech a připomínkách směrovaných na obchodníky. Primární klíč je zde umělý atribut *id_vzkazy*. Cizí klíč *nick* směrovaný na tabulku *a_uzivatel* udává autora vzkazu. Vzkazy se objeví všem uživatelům, jež mají právo obchodník nebo administrátor.

5.1.22. Tabulka A_ZOBRAZENI

Obrázek 27 - Tabulka A_ZOBRAZENI

Tabulka *a_zobrazeni* je uměle vytvořená tabulka reprezentující vazbu M:N mezi tabulkami *a_sekce* a *a_film,* neboť jeden film může patřit do více sekcí a zároveň v jedné sekci může být více filmů. Oba atributy v této tabulce jsou cizí klíče na výše zmíněné tabulky a kombinace těchto dvou atributů tvoří dohromady složený primární klíč.

5.1.23. Tabulka A_ZOBRAZENIH

Obrázek 28 - Tabulka A_ZOBRAZENIH

Tabulka *a_zobrazenih* je uměle vytvořená tabulka reprezentující vazbu M:N mezi tabulkami *a_sekceh* a *a_hudba,* neboť jedno hudební album může patřit do více sekcí a zároveň v jedné sekci může být více hudebních alb. Oba atributy v této tabulce jsou cizí klíče na výše zmíněné tabulky a kombinace těchto dvou atributů tvoří dohromady složený primární klíč.

5.2. Indexy

V této aplikaci jsou použity implicitní indexy, jež se vytvoří automaticky s primárním klíčem vyjma tabulky *a_info*, jež nemá primární klíč. Dále byli vytvořeny

indexy u atributů podle kterých bude probíhat nejčastější vyhledávání, což jsou jména nebo názvy u tabulek s filmovými či hudebními médii a u tabulek herců, režisérů a hudebních skupin.

5.3. Uložené funkce a procedury

V aplikaci je uložena jediná procedura *vratit*, která se doplňuje s triggerem *vypujcit*. Jejím úkolem je aktualizace počtu médií na skladě při vrácení libovolné výpůjčky a má dva parametry. První je identifikace zboží, zda se jedná o film či hudební album a druhý je identifikace výpůjčky, která je vrácena.

```
create or replace PROCEDURE vratit
( tab IN VARCHAR2
, idecko IN NUMBER
) AS
s NUMBER;
BEGIN
  if tab = 'film' then
  select pocet into s from a_film where id_filmu = idecko;
  s:=s + 1;
  update a_film set pocet=s;
  end if;

  if tab = 'album' then
  select pocet into s from a_hudba where id_album = idecko;
  s:= s + 1;
  update a_hudba set pocet=s;
  end if;
END vratit;
```

V databázi jsou také uloženy dvě funkce. První funkce se jmenuje *pocet* a vrací počet uživatelů s konkrétním oprávněním, jež je jejím argumentem.

```
create or replace FUNCTION "POCET" (priv a_uzivatel.nazevprava%TYPE) RETURN NUMBER AS
s NUMBER;
BEGIN
  SELECT count(*) INTO s FROM a_uzivatel WHERE nazevprava= priv;
  RETURN s;
END POCET;
```

Druhá funkce se jmenuje *pocetm* a vrací počet médií na skladě. Argumentem je typ zboží, zda se jedná o hudební či filmové médium.

```
create or replace FUNCTION "POCETM" (druh in VARCHAR2) RETURN NUMBER AS
s NUMBER;
BEGIN
  if druh='f' then
   SELECT count(*) INTO s FROM a_film;
  end if;
  if druh='h' then
   SELECT count(*) INTO s FROM a_hudba;
  end if;
  RETURN s;
END POCETM;
```

6 Návrh aplikace

Vlastní aplikace internetové videopůjčovny musí být vyvíjena jako webová prezentace, jež splňuje určitá kritéria. Musí být uživatelsky přívětivá, vhodně zabezpečená, dynamicky se měnící a musí obsahovat všechny potřebné funkce, jež může uživatel použít při půjčování médií.

6.1. Použité technologie

6.1.1 HTML

Základním stavebním kamenem je značkovací jazyk HTML (HyperText Markup Language), jež obsahuje základní grafické elementy webové stránky jako jsou tabulky, seznamy, obrázky, odkazy a mnoho dalších. Kostra aplikace a vizuální elementy budou napsány v jazyce HTML 4.01 Strict. HTML pracuje na straně klienta a dají se v něm vytvořit pouze statické stránky.

6.1.2 CSS

Jelikož bude aplikace půjčovny obsahovat značné množství webových stránek, je potřeba udržovat nastavení všech jejich grafických elementů v externím souboru oddělené od definic těchto elementů. Pro hromadnou úpravu vizuálního stylu celé webové prezentace je pak potřeba pouze pár úprav v tomto externím souboru. Pro tento účel slouží jazyk CSS (Cascading Style Sheets), který dovoluje v externím souboru nadefinovat libovolné styly a ty pak aplikovat na libovolný počet elementů v libovolném počtu webových stránek.

6.1.3 XML

XML (eXtensive Markup Language) je stejně jako HTML značkovací jazyk. Jeho tagy však nejsou předem definovány, definuje si je sám programátor podle své potřeby. Hlavní myšlenkou XML je zaznamenávat vlastní data spolu s informací, co daná data znamenají. XML lze tedy výhodně použít k exportu dat z dané webové stránky či aplikace. Například:

```
<film> <nazev_filmu> Avatar </nazev_filmu> <delka_filmu> 120 </delka_filmu> </film>
```

6.1.4 Javascript

Javasript je objektově orientovaný programovací jazyk pracující na straně klienta. S výhodou se proto dá použít pro ošetření formulářů, kterými do aplikace zadáváme data, neboť svojí činností na straně klienta nezatěžuje server. Přesto se na něj nelze stoprocentně spolehnout, neboť může být v internetovém prohlížeči zakázaný a formuláře by nebyly v tomto případě ošetřené. Proto bude tato činnost vyřešena pomocí PHP. Javascript se dá dále použít na animace, kontrolní a informační dialogy, směrování stránek dle historie prohlížeče a další podpůrné aplikace.

6.1.5 PHP

PHP dříve (Personal Home Page) dnes (Hypertext Preprocesor) je objektově orientovaný programovací jazyk pracující na straně serveru. Jeho syntaxe je velmi podobná programovacímu jazyku C, což zjednodušuje práci lidem, jež tento jazyk ovládají. Do webové stránky se může připisovat přímo mezi HTML tagy oddělený sekvencí **<?php** a **?>**. Výsledná data však klientovy posílá jako HTML kód. Webová stránka v jazyce PHP musí mít příponu **.php** a nemůže být ihned zobrazena pouze pomocí webového prohlížeče jako HTML stránka, nýbrž musí být umístěna v příslušném adresáři webového serveru jež má aktivní PHP modul.

Za pomocí PHP lze vytvořit plnohodnotné dynamické webové stránky a proto bude funkční část aplikace internetové půjčovny psaná v tomto jazyce spolu s ostatními výše zmíněnými jazyky.

Další důvod pro použití PHP je komunikace s databázovým serverem pomocí OCI_8 (Oracle Call Interface) funkcí.

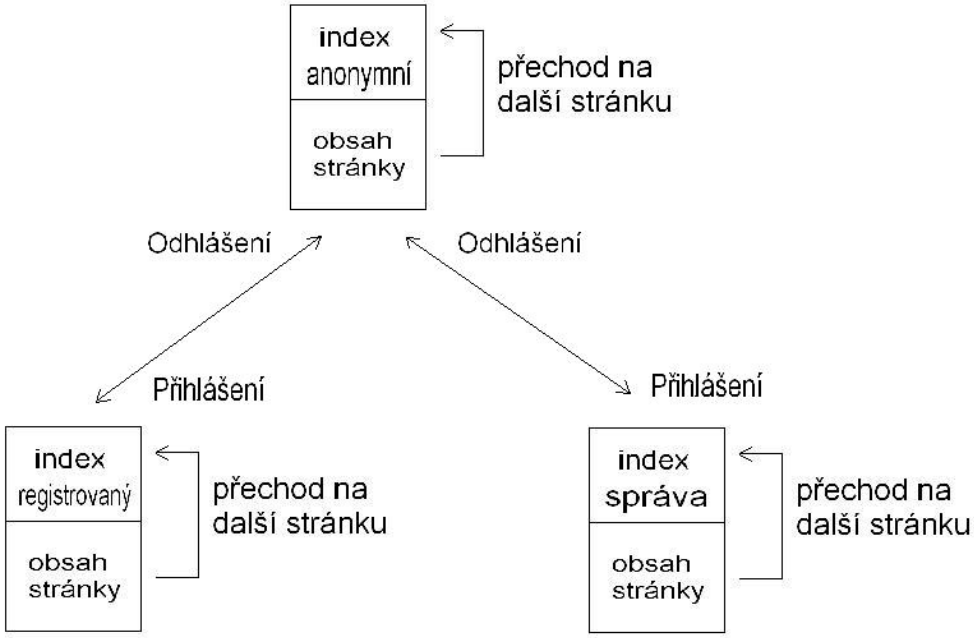
6.2. Návrh šablony webových stránek

Obrázek znázorňující rozložení částí webových stránek je přiložen v příloze G. Šablona webové stránky je tvořena z jednotlivých částí, které jsou uzavřeny do blokových elementů DIV, přičemž ke každému je v souboru kaskádových stylů přiřazen určitý styl. Veškeré vizuální prvky šablony jsou uzavřeny do hlavního DIV elementu třídy *.container* Tato část má pevně zvolenou šířku a délku má podle obsahu. Okolo tohoto elementu je ve zbytku stránky pouze textura, jejíž rozměry jsou dány podle

rozlišení monitoru počítače klienta. V horní části hlavního kontejneru je DIV element s třídou *.titleblock.* Ve spodní části je DIV s třídou *.footer* a na pravé straně DIV s třídou *.rightcontainer*. Ve zbytku hlavního kontejneru je DIV element s třídou *.content*.

Titleblock obsahuje název a logo půjčovny, název sekce dle práv a hlavní menu. Rightcontainer v sobě obsahuje dílčí DIV elementy třídy *.rightbox*, ve kterých se dle práv nachází rychlé vyhledávání, přihlašování či registrace do aplikace, základní informace o databázi, košík a administrační sekce. *Footer* obsahuje další informace na spodku stránky, které jsou vždy stejné. *Content* obsahuje vlastní obsah každé stránky.

6.3. Návrh uspořádání webových stránek



Obrázek 29 - Systém webových stránek

Možností, jakým stylem vytvářet webovou prezentaci je více. Pro internetovou půjčovnu byl vybrán styl **single_control_file**, který byl ještě poupraven kvůli existenci sekcí výhradně pro uživatele s určitým oprávněním. Samotný **single_control_file** je řešen tak, že se veškeré dění odehrává pouze na jedné kontrolní stránce, zpravidla index.php, které je vždy předán určitý parametr, podle kterého si kontrolní stránka naincluduje patřičný vnořený skript. Díky tomuto řešení je možné přidat prvky stránek, jež mají být stále stejné přímo do kontrolního souboru.

Podle výše zmíněného návrhu šablony bude DIV element třídy *.content* obsahovat includovaný vnořený skript a všechny zbylé elementy budou přímo v řídícím souboru. Dále takto řešeným způsobem odpadá problém s adresováním souborů, neboť veškeré dění probíhá stále v řídícím souboru.

Jak bylo již výše zmíněno, pro aplikaci internetové půjčovny bylo toto řešení poupraveno s ohledem na existenci sekcí pro různé druhy uživatelů. Stále je zde jeden hlavní řídící soubor index.php nicméně přihlášením uživatele do aplikace se veškeré dění přesouvá na jeden ze dvou vedlejších řídících souborů viz Obrázek 32. První je určen pro registrované uživatele a druhý pro správu systému. Oba pomocné řídící soubory jsou řešeny stejným způsobem jako hlavní, nicméně obsahují jiný obsah statických elementů a mají svůj seznam includovaných vnořených skriptů. Funkce kontrolního souboru může vypadat takto:

```
if(isset($_REQUEST['page']))
{
        $page=$_REQUEST['page'];
        if(check($page)==0){$page='hlavni';}
}
else
{
        $page='hlavni';
}
switch($page)
{
case 'hlavni': include 'basic/aktuality.php'; break;
case 'film': include 'basic/UkazFilm.php';} break;
case 'hudba': include 'basic/UkazHudbu.php';} break;
case 'hledani': include 'basic/Hledani.php';} break;
default: include 'basic/aktuality.php'; break;
}
```

Testuje se existence směrovací proměnné v globálních polích $_GET[], $_POST[] nebo $_COOKIE. Pokud proměnná v jednom z globálních polí existuje, zkopíruje se do lokální proměnné a otestuje se na výskyt nebezpečných znaků. Pokud proměnná v globálních polích neexistuje, tak je lokální proměnná vytvořena s hodnotou odkazu na hlavní stránku. Dále už je jen testuje obsah lokální proměnné na jehož základě kontrolní soubor naincluduje obsah stránky.

6.4. Návrh systému oprávnění

Jak bylo již výše zmíněno, jednotlivé možnosti aplikace a stránky, které může každý uživatel prohlížet, závisí na druhu oprávnění. Standartní řešení identifikace uživatele, je vytvoření uživatelského účtu a následné přihlašování. Úspěšným přihlášením se jednoznačně identifikuje uživatel, jež chce stránky prohlížet. Nyní je ale potřeba si tyto údaje pamatovat při přechodu na jiné stránky a zároveň zajistit, aby se k těmto údajům nedostala neoprávněná osoba. K tomuto problému se používá pole globálních proměnných zvané session, které si po registraci globálních proměnných pamatují jejich hodnotu dokud nebude manuálně smazána, nebude aplikace zavřena, případně po určitý časový interval v případě nečinnosti. V těchto proměnných je uchováváno přihlašovací jméno, zašifrované heslo a název oprávnění. V případě přihlášeného

uživatele se hodnota oprávnění kontroluje při vstupu na každou novou stránku a pokud hodnota neodpovídá, jsou proměnné vymazány a dojde k přesměrování na hlavní kontrolní soubor do sekce anonymních uživatelů.

7 Vývoj aplikace

Aplikace byla vyvíjena ve skriptovacím jazyce PHP za podpory HTML, CSS, XML a Javascryptu.

7.1. Umístění souborů aplikace

Hlavní adresář aplikace obsahuje hlavní kontrolní soubor *index.php*, sloužící pro anonymní uživatele. Dále se zde nachází soubor stylů *screenstyle.css,* v němž jsou nadefinované styly pro jednotlivé elementy všech webových stránek aplikace. Dále je zde skript *login.php,* který obsahuje php proměnné s informacemi, kterými se aplikace přihlašuje do databáze, soubor *.htaccess*, ve kterém je nastavení pro webový server, složka *Obr* obsahující obrázky k médiím uloženým v databázi, složky reCAPTCHA a tiny_mce, jež obsahují soubory potřebné k použití systému Captcha a WYSIWYG editoru. Dále zde jsou podadresáře aplikace *basic*, *uzivate*l a *admin* s dalšími skripty.

Adresář *basic* obsahuje vizuální skripty includované kontrolním souborem, skripty vykonávající různé akce a skripty obsahující funkci pro kontrolu nebezpečných znaků a funkci vyvolávající informační dialog pomocí javascriptu. Všechny skripty v tomto adresáři se buďto vztahují k anonymnímu uživateli nebo jsou tyto stránky společné pro všechny sekce.

Adresář *uzivatel* obsahuje kontrolní soubor *index.php* a vizuální skripty i skripty vykonávající určitou akci pro sekci registrovaných uživatelů. Tyto skripty se týkají objednávek, nákupního košíku a odesílání dotazů obchodníkům .

Adresář *admin* obsahuje kontrolní soubor *index.php* a vizuální skripty i skripty vykonávající určitou akci pro administrační sekci obchodníků a administrátorů. V tomto adresáři je ještě podadresář obsahující tyto skripty, jež týkají pouze manipulace se všemi druhy médií. Zbytek skriptů se zabývá správou uživatelů, aktualit, objednávek, vzkazů a upomínek.

7.2. Registrace uživatelů

Každý uživatel, jež chce využívat všechny funkce aplikace internetové půjčovny musí být zaregistrován. Registrace ho proto nesmí odradit od dalšího užívání aplikace. Neměla by od uživatelů vyžadovat zbytečné údaje, které nejsou pro objednávání zboží a další akce potřebné.

Registrace aplikace DVD půjčovny probíhá v přehledném formuláři. Uživatel musí zadat unikátní přihlašovací jméno, heslo, unikátní e-mail a svojí adresu, jež bude použita jako dodací adresa pro jeho objednávky. Kromě ulice musí být všechny položky

vyplněné a mají uvedenou minimální a maximální délku. U položky e-mail je ještě kontrolováno, zda zadaná hodnota obsahuje znak zavináče. Po vyplnění údajů je potřeba opsat dvě náhodně vygenerovaná slova do příslušné kolonky jako ověření, že je uživatel člověk a nikoliv spam robot.

Maximální délka řetězce v jednotlivých kolonkách je kontrolována nastavením maximálního možného počtu znaků přímo v příslušném vstupním poli. Kontrola ostatních podmínek, jako je minimální počet znaků, absence nebezpečných znaků, platný formát a unikátnost některých z řetězců je kontrolován v php skriptu a po splnění všech podmínek se zde provede zápis uživatele do databáze a proběhne přesměrování na hlavní stránku s upozorněním, že registrace proběhla úspěšně. V opačném případě proběhne přesměrování na registrační formulář s výpisem chyb jež nastaly.

Všechny údaje kromě uživatelského jména mohou být kdykoliv změněny v sekci *správa uživatelského účtu.*

7.3. Vyhledávání

Vyhledávání je jedna z nejdůležitějších funkcí DVD půjčovny. Předpokládá se, že bude velmi často využívána. Funkce vyhledávače by měla být snadno dostupná a umožnit vyhledání média i bez přesné znalosti jeho názvu z jeho části a nebo podle jiných parametrů.

V pravém horním rohu stránky aplikace se nachází dialog pro rychlé vyhledávání. Zde se zadává pouze název hledaného média či pouze jeho část. Po kliknutí na tlačítko *vyhledat* se zobrazí stránka s výsledky vyhledávání a zobrazí se shody s filmy, hudebními alby, hudebními skupinami, režiséry i herci.

Rozšířené vyhledávání je jedna z položek hlavního menu. V prvním dialogu si uživatel vybere, zda vyhledává film nebo hudební album a na základě toho vyplní patřičné kolonky. Atributy jako je herec, režisér nebo hudební skupina si uživatel pouze vybírá z příslušných možností. Jsou mu nabídnuty všechny záznamy těchto položek. U každého atributu lze vybrat možnost *libovolný/á* nebo příslušné pole nevyplnit.

Funkce rychlého vyhledávače je realizována ošetřením hledaného řetězce před nebezpečnými znaky a přidáním na jeho začátek i konec znaku „%“, jež v následném SQL dotazu zajistí vyhledání záznamu, kde je hledaný název pouze podřetězec názvu skutečného. SQL dotaz rychlého vyhledávání vypadá následovně:

```
SELECT * FROM a_film WHERE nazev_cesky LIKE '$nazev'
```

Tento dotaz se následovně použije i na ostatní tabulky v nichž jsou další média k vyhledání. Funkce rozšířeného vyhledávače spočívá v ošetření zadaného názvu před nebezpečnými znaky, přidání znaku „%“ před a za zadaný název, ošetření zadaného rozsahu let v němž dané médium vzniklo, aby obsahoval pouze číslice a pokud nebyl nějaký z parametrů vyplněn je prázdný řetězec nahrazen pouze znakem „%“, jež v následném SQL dotazu zaručí, že podmínce vyhoví jakýkoliv řádek tabulky.

Parametry u nichž uživatel pouze vybírá jednu z možností jsou dále předávány jako ID (Primární klíče) těchto záznamů. Vyhledávací SQL dotaz vypadá následovně:

```
SELECT DISTINCT a_film.id_filmu,nazev_cesky,nazev_originalni FROM a_film, a_hrani, a_zobrazeni
WHERE
(a_film.nazev_cesky LIKE '$nazev' OR a_film.nazev_originalni LIKE '$nazev')
AND a_film.id_rezisera LIKE '$idrez' AND a_hrani.id_herce LIKE '$idher' AND a_hrani.id_filmu =
a_film.id_filmu AND a_film.rok > '$r1' AND a_film.rok < '$r2'
AND a_zobrazeni.id_sekce LIKE '$idsek' AND a_zobrazeni.id_filmu = a_film.id_filmu
```

7.4. Objednávání

Vybírání a následné objednání zboží je nejdůležitější funkcí DVD půjčovny. Uživatel by měl mít možnost zboží přidat do košíku, odebrat ho z něj a samozřejmě i zboží z košíku objednat k vypůjčení.

Zboží lze objednat pouze uživateli, jež je přihlášen a pouze, když je dané zboží na skladě tzn. pokud je počet zboží větší než nula. Uživateli se po úspěšném přihlášení do systému zobrazí v pravé boční liště sekce *košík*, kde se podle vybírání zboží přidávají jednotlivé položky. U každé položky je možnost odebrání z košíku a pod nimi tlačítko *objednat*, jež uživatele přesune na stránku objednávky. Zde je seznam zboží k objednání, celková cena objednávky a možnost každé zboží z objednávky odstranit. Dále zde jsou kontaktní údaje, které jsou automaticky vloženy z uživatelova profilu a možnost napsat poznámku k objednávce obchodníkovi, jež ji bude vyřizovat.

V aplikaci je jednotlivá položka v košíku realizovaná jako instance objektu, který se skládá z id hudebního alba, id filmu a názvu položky. Jeden z prvních dvou atributů bude vždy obsahovat hodnotu 0. Košík je následně realizován jako dynamické pole těchto objektů. Jelikož se košík a položky v něm musí objevovat na každé stránce v sekci registrovaných uživatelů, musí být toto pole v session *$_SESSION["kos"]* a objektová struktura každé položky musí být ještě rozložena na proud znaků, aby se dala v session přenášet.

```
$tmp=new Tkosik;
$pole = array();
reset($_SESSION['kos']);
while(list($i,$entita) = each($_SESSION['kos'])){
$tmp=unserialize($entita);
$pole[]=$tmp;}
unset($pole[$inx]); //$inx je index položky jež má být smazána
unset($_SESSION['kos']);
$_SESSION['kos'] = array();
while(list($i,$tmp) = each($pole)){
$entita=serialize($tmp);
$_SESSION['kos'][]=$entita;}
```

Odebírání konkrétní položky z košíku spočívá v tom, že se celý košík zkopíruje do lokálního pole, jeho jednotlivé položky se složí do původní struktury, konkrétní položka, jež má být odebrána je z pole vymazána a obsah lokálního pole je zpět rozložen a přesunut zpátky do session *$_SESSION["kos"]*.

Vlastní objednání zboží spočívá v přesunu položek z *$_SESSION["kos"]* do lokálního pole, kontrole, zda je každá položka skutečně na skladě a vytvoření nového záznamu v tabulce *a_vypujcka* se stavem *ceka,* a vytvoření nových záznamů do tabulky *a_kosik,* jež každý spustí trigger *vypujcit*, který zajistí snížení počtu jednotlivé položky a přepočítá celkovou cenu objednávky.

7.5. Správa výpůjček, upomínek a vzkazů

Další důležitou funkcí DVD půjčovny je upozornění v případě, že nějaká výpůjčka přesáhla stanovenou dobu, během které měla být vrácena. Upozorněn by měl být obchodník a následně i zákazník, jež má dané média vypůjčené.

Správa výpůjček, upomínek a vzkazů se nachází v administrátorské sekci pod podsekce *správa obchodu*. Nachází se zde tabulka vzkazů, na které lze reagovat a poté je smazat a tabulka s objednávkami, které jsou setříděné podle stavu v pořadí: čekající, půjčené a vrácené a dále podle data vytvoření. Čekající výpůjčku lze potvrdit, čímž se její stav změní na půjčeno a automaticky se vygeneruje nová stránka / záložka s fakturou určenou pro tisk. Půjčenou výpůjčku lze vrátit, čímž se její stav změní na vráceno a následně se aktualizuje počet kusů jednotlivých položek výpůjčky na skladě. Vrácenou výpůjčku lze smazat každou zvlášť i hromadně všechny, které mají stav vráceno. Výpůjčky se jednotlivým uživatelům zobrazují v jejich profilu.

Dále se ve správě obchodu nachází podsekce správa upomínek, do které když obchodník vstoupí, tak se automaticky zkontrolují všechny aktivní výpůjčky, ke kterým nebyla dosud vygenerována žádná upomínka a pokud některá z nich přesahuje dobu 30 dnů, automaticky je k ní vytvořena upomínka ve stavu čeká. Dále se upomínka zobrazí danému uživateli v jeho profilu. Tuto upomínku může obchodník aktivovat, čímž se přesune na předvyplněný mailový formulář s možností upravit automatické informace a odeslat uživateli mail o upomínce. Po vrácení výpůjčky, která obsahuje aktivní upomínku se tato upomínka nastaví do stavu vráceno. SQL dotaz pro automatickou kontrolu upomínek vypadá následovně:

```
SELECT * FROM a_vypujcka WHERE id_vypujcka in (SELECT id_vypujcka FROM a_vypujcka MINUS SELECT id_vypujcky FROM a_upominky) AND EXTRACT (DAY from (SYSDATE-cas)) > 30 AND a_vypujcka.stav='pujceno'
```

7.6. Správa médií

Nedílnou součástí DVD půjčovny je rozsáhlá možnost administrace zboží, jež nabízí. Tato administrace by měla být patřičně ošetřena, aby se při mazání položky

z databázové tabulky také smazaly reference na tento záznam v jiných tabulkách a stanovit, kdy je možné záznam bezpečně smazat.

Správa médií se nachází v administrátorské sekci a umožňuje přidávat, editovat a odebírat všechny filmy, hudební alba, hudební skupiny, režiséry a herce. Přiřazení hudebních skupin, herců a režisérů filmům a hudebním albům probíhá tak, že je například při přidávání hudebního alba nutno nejprve přidat patřičnou hudební skupinu, a poté ji při přidávání hudebního alba vybrat z příslušné nabídky. Stejný princip je při přidávání herců a režiséra k filmu. Při odebírání platí opačný postup. Například režiséra lze odebrat z databáze pouze, pokud v databázi neexistuje žádný film, ke kterému je daný režisér přiřazen. Dále nelze smazat hudební alba a filmy, k nimž se vztahuje aktivní výpůjčka, aby nevznikla chyba v systému a pokud jsou tyto média mazány jsou smazány i vrácené výpůjčky.

7.7. Správa uživatelů

V každém soběstačném systému by měl být alespoň jeden administrátor, který má kromě všech možností, které mají uživatelé s ostatními oprávněními i právo spravovat ostatní uživatele v systému.

Správa uživatelů se nachází s administrátorské sekci a umožňuje mazat tyto ostatní registrované uživatele a měnit jim práva. Přístup sem má pouze uživatel s administrátorským oprávněním. Nachází se zde tabulka uživatelů, kterou lze seřadit podle: času registrace, oprávnění, abecedně dle přihlašovacího jména a podle doby od posledního přihlášení do systému. Nelze však smazat uživatele jež mají v systému aktivní výpůjčku. Dále po smazaní uživatele dojde ke smazání všech jeho vrácených výpůjček a po odebrání obchodníka či administrátora dojde ke smazání všech jím napsaných aktualit

7.8. Správa aktualit a informací

Důležitá funkce DVD půjčovny je informování všech uživatelů o různých změnách či novinkách. Tyto informace by měli být přehledně umístěné, správně seřazené a v případě většího množství také uskupené do jednotlivých stránek.

Správa aktualit se nachází v administrátorské sekci a umožňuje obchodníků a administrátorům informovat anonymní i přihlášené uživatele o nových skutečnostech, případě změnách. Tyto informace se zobrazují na hlavní stránce aplikace a řadí se do stránek po deseti. Tyto informace se přidávají pomocí přehledného WYSIWYG editoru *Tiny_mce*. Tyto informace lze smazat a jejich autor nebo administrátor je může změnit.

7.9. Prohlížení a export médií

Vlastní seznam filmů nebo hudebních alb lze zobrazit z hlavního horního menu pod položkou *filmy* nebo *hudba*. Po kliknutí na jednu z položek se zobrazí seznam všech médií společně se seznamem sekcí, do kterých jednotlivé média patří. Po kliknutí na jednotlivé sekce se seznam médií vyfiltruje pouze na ty, které do dané sekce patří.

Jedno médium může patřit do více sekcí. Ve spodní části stránky je také možnost *export do xml*. Po kliknutí lze zobrazit nebo uložit seznam médií v XML struktuře.

8 Závěr

Cílem práce bylo vytvořit webovou aplikaci pro internetovou DVD půjčovnu s využitím databáze Oracle. Aplikace byla naprogramována jazykem PHP, který obstarává dynamičnost webových stránek, logiku aplikace, výpis dat a komunikaci s databázových serverem. V aplikaci jsou použity ještě jiné pomocné technologie. Přestože aplikace je vcelku přehledně napsaná a plně funkční, ještě by lze bylo výhodné přesunou ucelené bloky kódu, které vykonávají specifickou funkci do php funkcí, které by byly umístěny v jednom php skriptu, čímž by došlo k redukci počtu souborů v aplikaci. Zbytek php kódu by se stal přehlednějším a případné funkční úpravy by se týkaly pouze jednoho konkrétního skriptu. Dále by bylo vhodné použít ve větším měřítku technologii javascriptu na různé kontroly na straně klienta, kvalitnější tabulky s rozšířenou možností třídění údajů a našeptávač u některých vstupních polí. Další a pravděpodobně nejlepší způsob jak zvýšit kvalitu a přehlednost aplikace by bylo použití některého z frameworků, který by aplikaci rozdělil na nezávislé celky pomocí archi-tektury MVC.

K této aplikaci by z funkčního hlediska bylo dobré přidat možnost uživatelů hodnotit filmy a také je komentovat. Z hlediska půjčovny zde chybí také systém rezervací aktuálně vypůjčených filmů.

Nakonec se podařilo napsat aplikaci, která zvládá všechny standartní funkce jež k DVD půjčovně patří s rozsáhlými administračními možnostmi uživatelů, filmových a hudebních médií, herců, režisérů, hudebních skupin. Je zde možná pohodlná komunikace mezi uživatelem a obchodníkem. Aplikace od uživatelů ani obchodníků nevyžaduje žádné odborné znalosti a všechny funkce jsou intuitivní a snadno dostupné.

Seznam použité literatury

[1] Dlouhý, R. *PHP v příkladech*. Computer Media, 2007

[2] Castagnetto, J. a kol. *Programujeme PHP profesionálně*. Computer Press, 2004.

[3] Oppel, A. *Databáze bez předchozích znalostí*. Computer Press, 2006

[4] Lacko, L. Oracle - *správa, programování a použití databázového systému*. Computer Press, 2007.

[5] Kofler, M. *Mistrovství v MySQL 5 Kompletní průvodce webového vývojáře*. Computer Press, 2007

[6] Krch, D. *Oracle Database Express Edition* [online]. 2007 [cit. 2010-23-04]. Dostupný z WWW: <http://www.linuxexpres.cz/business/oracledatabase-express-edition>.

[7] *PHP* [online]. 2001-2009 [cit. 2009-11-09]. Dostupný z WWW: <http://www.php.net/>.

[8] *Databáze standartu SQL* [online]. [cit. 2010-04-20]. 1998. Dostupný z WWW: <http://www.penguin.cz/noviny/?id=chip/index/>.

[9] *Oracle database* [online]. [cit. 2010-04-23]. 2001-2010. Dostupný z WWW: <http://www.oracle.com/database/>.

[10] *Wikipedie, otevřená encyklopedie*[online]. [cit. 2010-04-23]. 2001-2010. Dostupný z WWW: <http://cs.wikipedia.org/>.

[11] *MYSQL* [online]. [cit. 2010-04-23]. 2001-2010. Dostupný z WWW: <http://www.mysql.com/>.

[12] Západočeská univerzita v Plzni, katedra informatiky a výpočetní techniky. P*okročilý SELECT v SŘBD Oracle* [online]. [cit. 2010-04-23]. 5.6.2008. Dostupný z WWW: <http://www.kiv.zcu.cz/~zima/vyuka/db2/sql92-02.html/>.

Příloha A – Instalace

Aplikace je napsaná ve skriptovacím jazyku PHP. Pro instalaci aplikace je tedy nutné mít nainstalován webový server Apache s PHP. Dále je nutné nainstalovat a nakonfigurovat databázový systém Oracle. Všechny balíky, které jsou nutné pro běh aplikace, lze stáhnout zdarma, viz níže.

Webový server Apache + PHP

http://www.wampserver.com/en/

Databázový server Oracle 10g Express Edition

http://www.oracle.com/technology/software/products/database/xe/index.html

Konfigurace databázového serveru

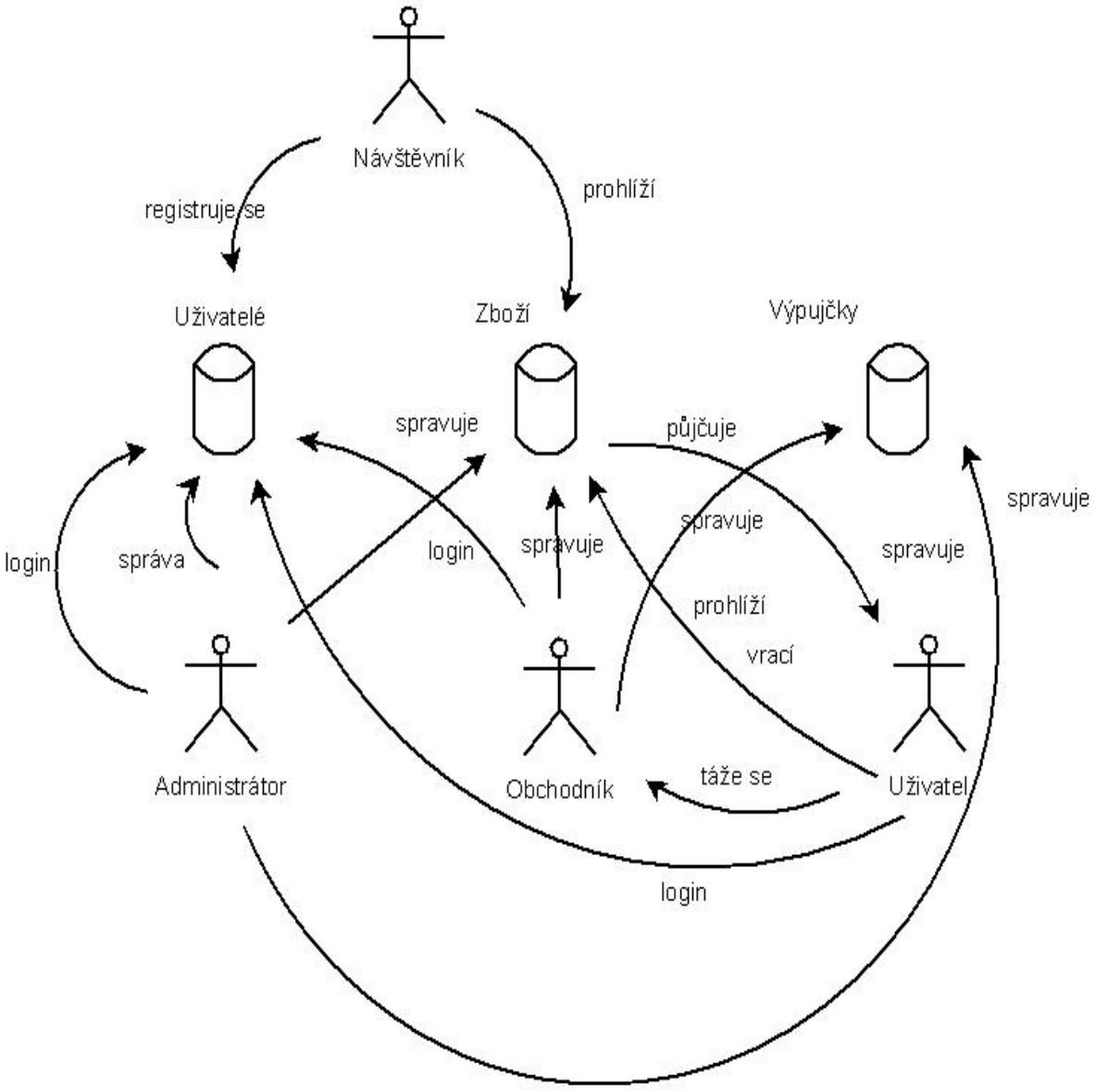
Po instalaci databázového serveru je potřeba vytvořit v databázi účet, který bude mít práva pro tvorbu SQL objektů. Přihlašovací údaje tohoto účtu je nutné zanést do konfiguračního souboru aplikace login.php. V tomto souboru je třeba přiřadit dané údaje do PHP proměnných. *$conect* obsahuje informace o umístění databázového serveru. *$user* obsahuje informace o názvu uživatelského účtu v databázi a *$password* obsahuje informace o heslu k tomuto účtu.

Dále je třeba se za tohoto databázového uživatele přihlásit do databáze a spustit přiložené skripty, které v databázi vytvoří tabulky, jejich data a další objekty.

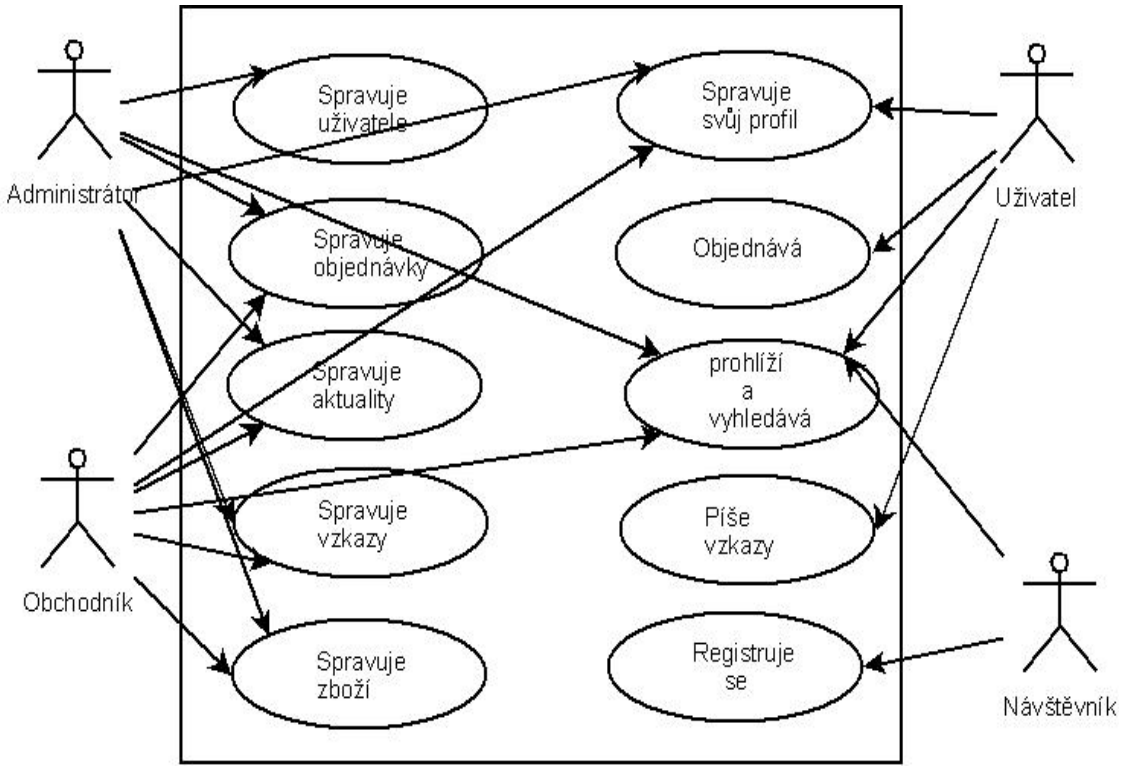
Kopírování aplikace

Posledním krokem před spuštěním samotné aplikace je vzít adresář s aplikací pojmenovanou např. DVD a zkopírovat jej do adresáře WWW v kořenovém adresáři webového serveru WAMP. Dále už jen stačí napsat do adresy webového prohlížeče localhost, případně localhost:XX, kde XX je číslo portu pokud apache neběží na portu standartně nastaveným instalací a kliknout na položku DVD, ve kterém server automaticky vyhledá spouštěcí skript (index.php) a spustí jej.

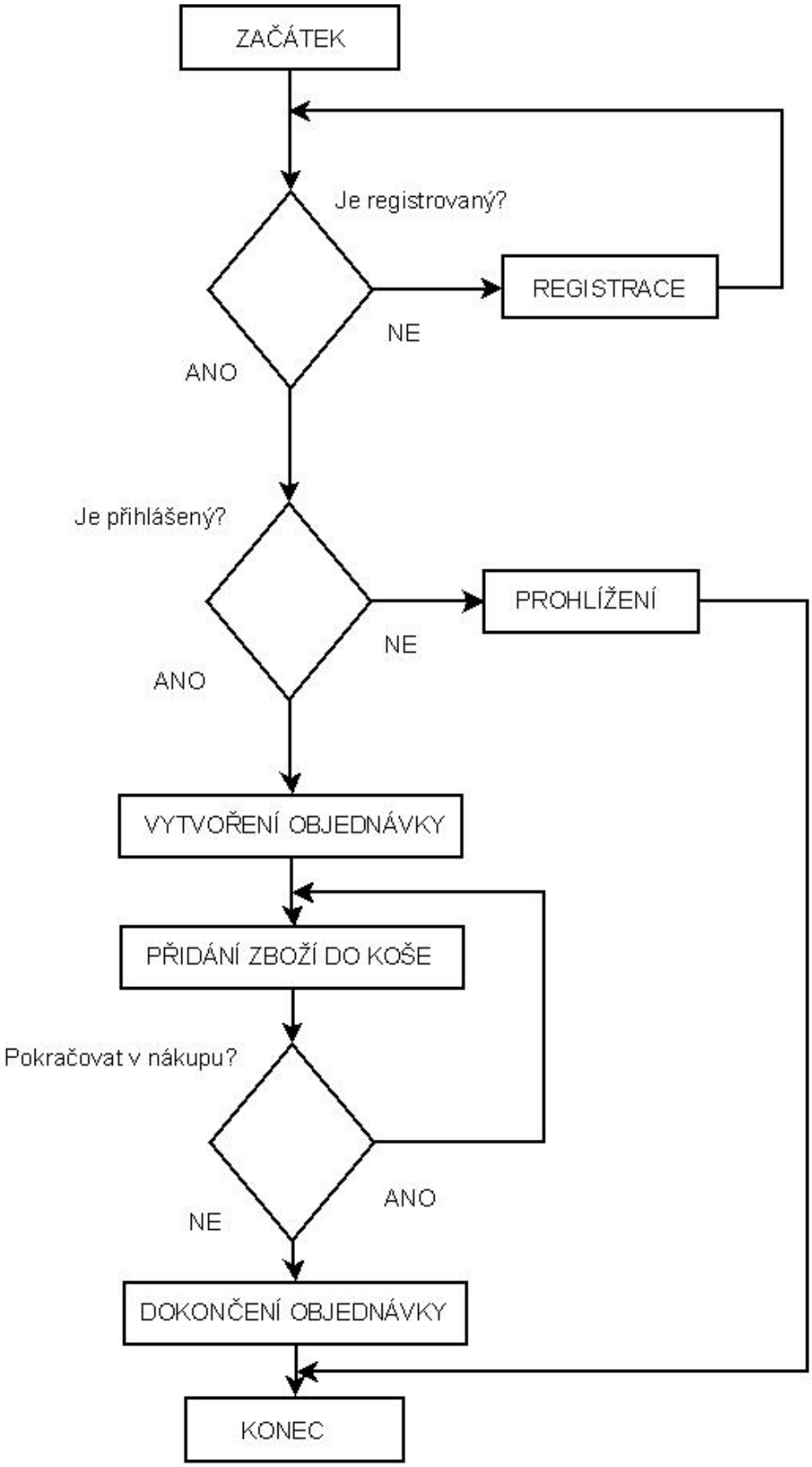
Příloha B – Rich picture diagram



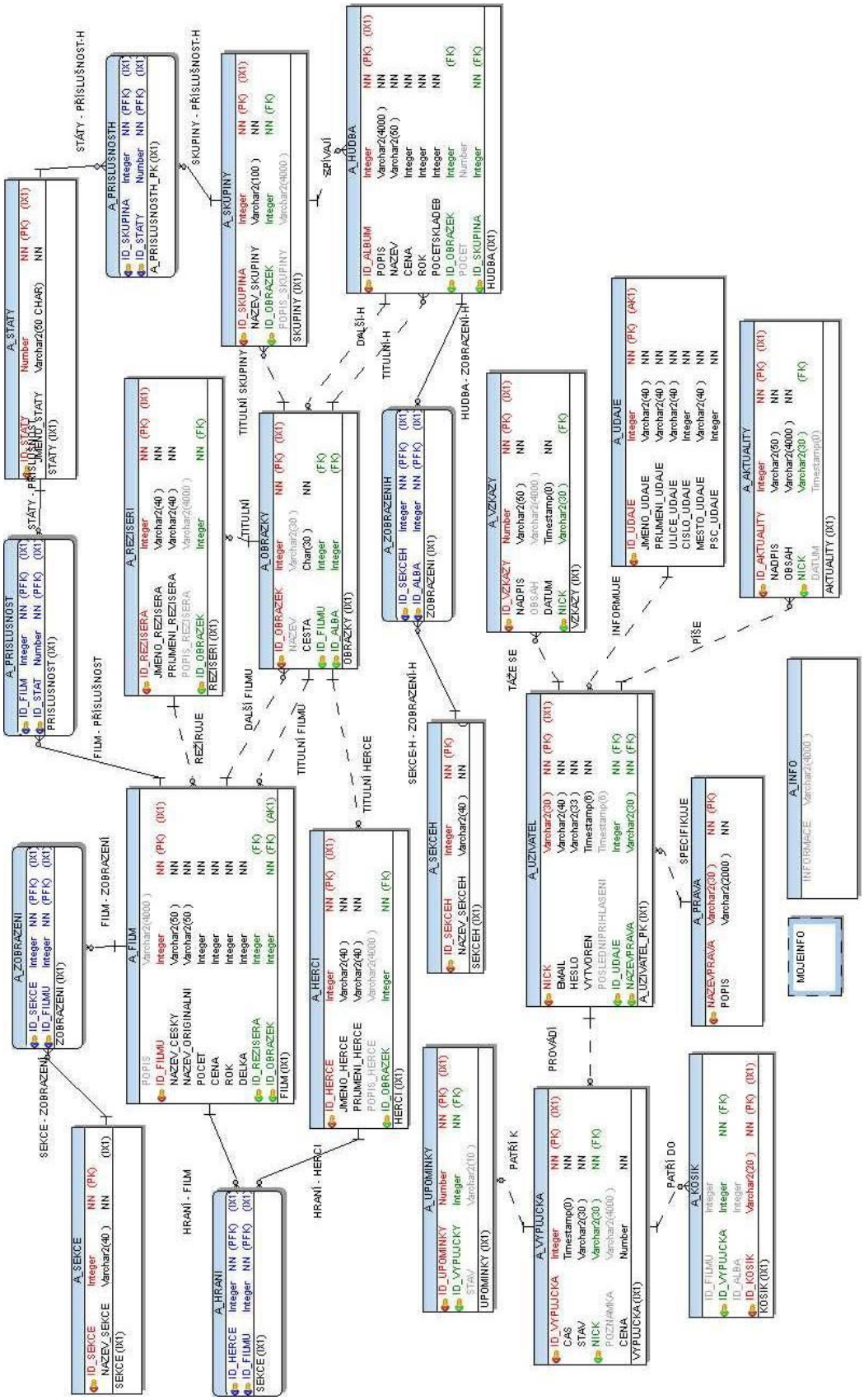
Příloha C – UML usecase diagram



Příloha D – UML activity diagram



Příloha E – ER-diagram



Příloha F – Trigger „vypujcit“

```
create or replace TRIGGER "VYPUJCIT"
before insert on a_kosik
for each row
declare
x number;
c1 number;
c2 number;
begin
 if :NEW.id_filmu > 0 then
   select POCET INTO x from a_film where id_filmu=:NEW.id_filmu;
   update a_film set pocet=x-1 where id_filmu=:NEW.id_filmu;

   select CENA INTO c1 from a_vypujcka where id_vypujcka = :NEW.id_vypujcka;
   select CENA INTO c2 from a_film where id_filmu=:NEW.id_filmu;
   update a_vypujcka set cena=(c1+c2) where id_vypujcka=:NEW.id_vypujcka;
end if;

 if :NEW.id_alba > 0 then
   select POCET INTO x from a_hudba where id_album=:NEW.id_alba;
   update a_hudba set pocet=x-1 where id_album=:NEW.id_alba;

   select CENA INTO c1 from a_vypujcka where id_vypujcka= :NEW.id_vypujcka;
   select CENA INTO c2 from a_hudba where id_album=:NEW.id_alba;
   update a_vypujcka set cena=(c1+c2) where id_vypujcka=:NEW.id_vypujcka;
end if;
end;
```

Příloha G – Šablona webových stránek