TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N2612 - Elektrotechnika a informatika
Studijní obor: 1802T007 - Informační technologie

UNIVERZÁLNÍ WEBOVÁ APLIKACE NA PŘEVOD DAT MEZI DATABÁZEMI

UNIVERSAL WEB APLICATION FOR TRANSFERING DATA BETWEEN DATABASES

Diplomová práce

Autor: Bc. Jan Kahoun
Vedoucí diplomové práce: Doc. RNDr. Pavel Satrapa, Ph. D
Konzultant: Dis. Martin Hujsl

V Liberci 25. 5. 2009

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií
Ústav nových technologií a aplikované informatiky
Akademický rok: **2007/2008**

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno a příjmení:	Jan Kahoun
Studijní program:	M 2612 - Elektrotechnika a informatika
Studijní obor:	Informační technologie
Název tématu:	**Univerzální webová aplikace na převod dat mezi databázemi**

Zásady pro vypracování:

1. Vytvořte univerzální webovou aplikaci za pomoci programovacího jazyka PHP, která bude umožňovat převod dat mezi databázemi s různou strukturou nebo databázemi různého typu.
2. Navrhněte systém tak, aby umožňoval převod dat mezi databázemi MySQL, PostgreSQL a FirebirdSQL, včetně jednoduché možnosti rozšíření o další databáze bez nutnosti zásahu do zdrojových kódů aplikace.
3. Vytvořte v systému mechanismy, které zajistí výstup a vstup dat v různých formátech (CSV, XSL, XML).
4. Ověřte funkčnost aplikace vytvořením demonstračních příkladů.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

ABSTRAKT

Tato práce se v první části zaměřuje na potřebnou teorii o databázových systémech a možném převodu dat mezi různými typy. Hlavní náplní je návrh a implementace webové aplikace, která umožní připojení k různým typům databázových systémů, včetně možnosti rozšíření o nové typy, kopírování dat a nabídne mechanismy, které zajistí výstup a vstup dat v různých formátech. Na závěr prezentuje dosažené výsledky vytvořenými demonstračními příklady.

ABSTRACT

The first part of this presentation is mainly about needful theory of database systems and about possibly transfering these data between different types. The project and implementation web application is the main contents which is able to link, to a different types of database systems, including possibility of expansion a new types, copying data and mechanism, which provides to entry and way out of these data in a different size. In the last step of this presentation we can see effects generated from demonstrational exemplars.

Klíčová slova

databáze, SQL, univerzální, kopírování, import, export

Keywords

database, SQL, universal, copying, import, export

OBSAH

SEZNAM ZKRATEK

ACID - Atomicity, Consistency, Isolation, Durability

AJAX - Asynchronous JavaScript and XML

ANSI - American National Standards Institute

API - Application programming interface

CMS - Content Management System

CSS - Cascading Style Sheets

DB - Databáze

DBMS - Database managment system

DFD - Data Flow Diagram

DOM - Document Object Model

DTD - Document Type Definition

ERD - Entity Relationship Diagram

HTML - Hypertext Markup Language

MySQL - My Structured Query Language

ODBC - Open DataBase Connectivity Standard

OOP - Objektově orientované programování

PHP - PHP: Hypertext Preprocessor

RMD - Relační model dat

SQL - Structured Query Language

SGML - Standard Generalized Markup Language

UML - Unified Modeling Language

W3C - World Wide Web Consortium

WWW - World Wide Web

XHTML - eXtensible HyperText Markup Language

XML - eXtensible Markup Language

SEZNAM OBRÁZKŮ

1. ÚVOD

V dnešním počítačovém světě se s databázemi můžeme setkat téměř všude. Jejich nasazování a používání je natolik rozšířené, že zájem o nástroje, které s nimi dokážou pracovat ať už v podobě on-line nebo stolních aplikací, je obrovský. Požadavkem poslední doby jsou pak unifikované nástroje, neboť máme k dispozici mnoho různých databázových systémů (DBMS[1]) s rozdílnými vlastnostmi. Díky rozličnostem DBMS se objevuje problém převodu dat mezi nimi, který nemusí být vždy jednoduchý. Databáze mohou být nejen různé struktury ale i různého typu. S touto problematikou úzce souvisí také problém importování a exportování dat.

Podobný problém řeší i firma CLWEB. Ta má své vlastní webové aplikace, především CMS systémy a elektronický obchod, které nabízí zákazníkům. Ti většinou přicházejí, s požadavkem přechodu ze stávajícího systému na systém firmy CLWEB, což obnáší přenos dat z jedné databáze do druhé. V tomto okamžiku vždy dochází k vytvoření speciálního PHP skriptu, řešícího konkrétní situaci. Řešení je ovšem velice neefektivní, především z časového hlediska a proto by společnost ráda měla aplikaci, která by jim toto usnadnila.

Mým úkolem je proto vyvinout univerzální aplikaci zpřístupňující unifikované nástroje pro práci s rozličnými databázemi, co se týče struktury nebo jejich typu. Aplikace by měla být uživatelsky přívětivá, přehledná, jednoduše ovladatelná a snadno rozšiřitelná, aby byl umožněn její další rozvoj.

[1] Běžně se, dle souvislosti, slovem databáze označují uložená data i DBMS.

2. POPIS PROBLÉMU, SPECIFIKACE CÍLE

2.1. CÍLE DIPLOMOVÉ PRÁCE

Základním cílem je zjednodušit práci při převodu dat, jelikož v současné době dochází vždy k vytvoření speciálního skriptu, řešícího konkrétní situaci.

Jak bylo v této kapitole napsáno, problematika databází je rozsáhlá a složitá. Z těchto důvodů není možné, aby výsledná aplikace pokrývala veškerou problematiku. Mohla by mít rozličné vlastnosti, obsahovat mnohé funkce, implementovat mnoho myšlenek a nápadů, ale to není jejím hlavním cílem. Měla by být jakýmsi *„základním kamenem"*, na kterém se bude moci dále stavět a který umožní další vývoj a rozvoj o nové funkcionality a vlastnosti dle požadavků, nápadů a myšlenek.

Celkového zjednodušení by se mělo dosáhnout splněním následujících definovaných cílů:

- vytvořit univerzální webovou aplikaci v jazyce PHP umožňující převod dat mezi databázemi MySQL, PostgreSQL a Firebird
- umožnit jednoduché rozšíření o další typy databází bez nutnosti zásahu do zdrojových kódů aplikace
- zpřístupnit nástroje pro výstup a vstup dat ve formátech CSV, XLS a XML
- ověřit funkčnost aplikace vytvořením ukázkových příkladů

2.2. DEFINICE PROBLÉMU

Problematika relačních DBMS je velmi rozsáhlá, proto se tato kapitola bude věnovat pouze potřebné teorii, která bude posléze využita k návrhu a realizaci aplikace. Každý DBMS je jiný, proto se na konci některých podkapitol zmiňuji o konkrétních rozdílech mezi MySQL, PostgreSQL a Firebird DBMS.

2.2.1. Databáze

Nejčastější použití počítačů je v oblasti zpracování dat. Rozsáhlejší systémy, které zpracovávají data, se nazývají tzv. *informační systémy*. Slouží ke sběru, uchování, vyhledání a zpracování dat. Tato data je nutné někde uchovávat a k tomu slouží většinou DBMS, které mají několik výhod oproti datovým souborům:

- jsou poměrně jednoduché na přístup k datům
- zachovávají referenční integritu dat (primární a cizí klíče, triggery, omezení)
- podporují transakční zpracování dat
- řeší problém současného přístupu více uživatelů
- jsou bezpečné
- při dobrém návrhu nenastává problém redundance dat

V dnešní době jsou nejčastěji používané relační databáze postavené na relačním modelu dat (RMD).

Základy relačního modelu dat

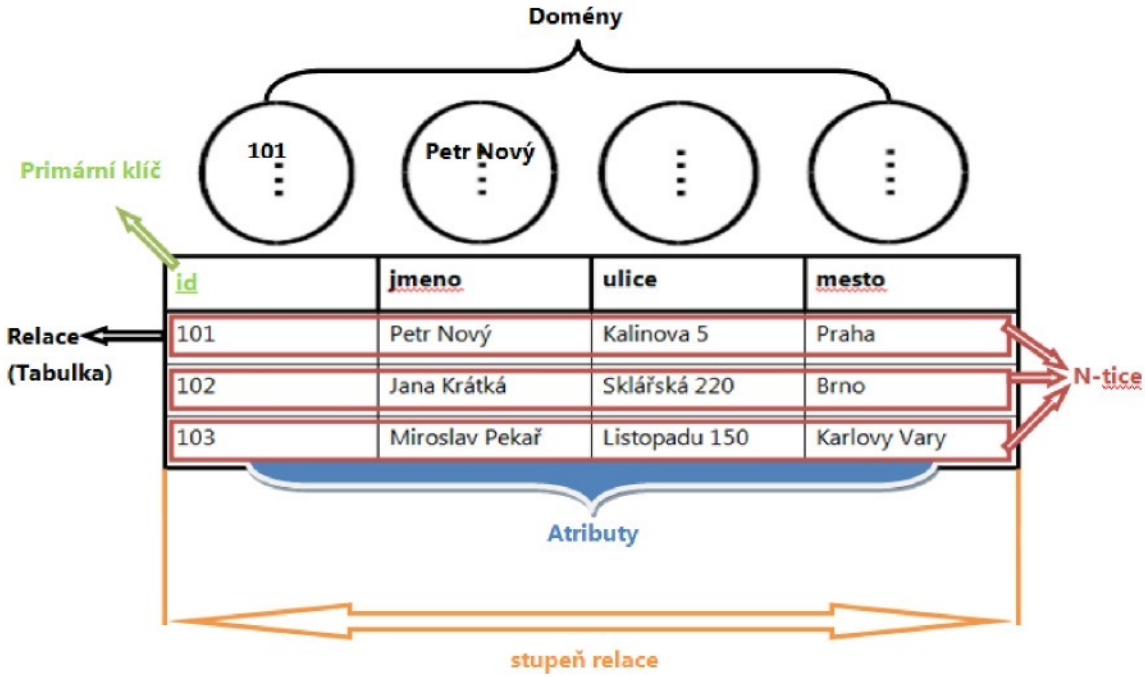
Relační model navrhl a jeho pravidla publikoval Dr. E. F. Codd. Jedná se o datový model založený na predikátové logice a teorii množin, který také definuje způsob reprezentace dat, způsob jejich ochrany (integritní omezení) a možné operace nad daty. Mezi základní ideje patří:

- RMD důsledně odděluje data, která jsou chápána jako relace, od jejich implementace.
- Přístup k datům je symetrický. Při manipulaci s daty se nezajímáme o přístupové metody k datům.
- Pro práci s daty máme k dispozici relační kalkul a algebru (matematické aparáty, jimiž lze popsat sémantiku (význam konstrukcí) relačních jazyků).

- Pro omezení redundance dat máme k dispozici pojmy pro normalizaci relací, jejímiž výsledky jsou vhodně navržené databázové struktury.

Relační struktura dat

Relační struktura dat je znázorněna na obrázku č. 1.



obrázek č. 1 - relační struktura dat

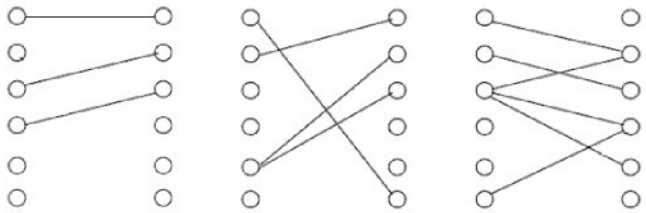
Jednoduchý popis znázorněné struktury:

- **Relace** – základní abstraktní pojem relačního modelu (tabulka je její znázornění)
- **Atributy** – sloupce v relaci
- **N-tice** – řádky (záznamy) v relaci, kde N určuje stupeň relace
- **Stupeň relace** – počet atributů relace
- **Domény** - pojmenované množiny hodnot téhož typu
- **Primární klíč** - kombinace jednoho a více atributů, které jsou unikátní pro daný záznam
- **Cizí klíč** – kombinace jednoho a více atributů, které odkazují na primární klíč

Vztahy mezi tabulkami

Vztahy slouží ke svázání dat, která spolu souvisejí a jsou umístěna v různých tabulkách. Rozlišujeme tři vztahy (znázorněny na obrázku č. 2):

- **1:1** – například student x trvalé bydliště
- **1:N** – například studijní skupina x student
- **N:M** – například student x předměty



obrázek č. 2- vztahy mezi relacemi

Podmínky relačnosti tabulky

- všechny hodnoty v tabulce musí být elementární (nedělitelné)
- pozice sloupců je nevýznamná (pořadí lze měnit)
- pozice řádků je nevýznamná (pořadí lze měnit)
- obor hodnot sloupce musí být stejný
- každý sloupec musí být jednoznačně pojmenován
- každý řádek musí být jednoznačně rozlišen (primární, cizí klíč)

Operace s relacemi

S každou relací lze provádět dva základní druhy operací, relační a množinové:

Relační operace

- **selekce** – výběr požadovaných řádků
- **projekce** – výběr požadovaných atributů (sloupců)
- **spojení** – spojení dvou relací, při kterém vzniká třetí, která obsahuje všechny kombinace vyhovující zadané podmínce

Množinové operace

- **sjednocení** - spojení více atributů do jednoho

- **průnik** – výběr řádků, které mají společné hodnoty pro obě původní množiny řádků
- **rozdíl** - výběr řádků, které nemají společné hodnoty pro obě původní množiny řádků
- **kartézský součin** - kombinuje každý řádek z první množiny se všemi řádky z druhé množiny

2.2.2. Integritní omezení

Obecně tvoří integritní omezení (IO) mechanismy zajišťující, aby se do databáze dostala jen data, která tam patří a neztratila se data, která nemají. Pokud, jsou v databázi data vyhovující IO, říkáme, že je v konzistentním stavu. To znamená, že úpravou nebo smazáním se nám data neztratila, nepoškodila, a že v DB nejsou data, která tam nemají co dělat.

Rozeznáváme tři druhy integritních omezení, dle toho jak může dojít k jejímu porušení:

- **Entitní** – specifikace primárního klíče v tabulce
- **Doménové** – omezení sloupce na určitý datový typ, či rozsah hodnot
- **Referenční** – vztah dvou tabulek pomocí cizího klíče (atribut (skupinu atributů), které mají buďto hodnotu NULL, nebo hodnotu primárního klíče některého řádku z tabulky na které je cizí klíč závislý)

2.2.3. SQL

SQL je neprocedurální jazyk, nepopisuje jak pracovat s databází, ale co od ní uživatel potřebuje. Skládá se z několika částí:

- **DDL (Data Definition Language)** – jazyk pro definici struktury databáze
  - `CREATE` – vytváření nových objektů.
  - `ALTER` – změny existujících objektů.
  - `DROP` – odstraňování objektů.

- **DML (Data Manipulation Language)** – jazyk pro manipulaci s daty
  - `SELECT` – vybírá data z databáze, umožňuje výběr podmnožiny a řazení dat
  - `INSERT` – vkládá do databáze nová data
  - `UPDATE` – mění data v databázi (editace)
  - `MERGE`[2] – záznam se do tabulky vloží, pokud v tabulce neexistuje odpovídající klíč, nebo se záznam upraví
  - `DELETE` – odstraňuje data (záznamy) z databáze
  - `SHOW` - *příkaz pro zobrazení databáze, tabulek nebo záznamů*
- **DCL (Data Control Language)** – jazyk pro řízení transakcí a nastavování přístupových práv
  - `GRANT` – příkaz pro přidělení práv uživateli k určitým objektům
  - `REVOKE` – příkaz pro odejmutí práv uživateli
  - `BEGIN` – zahájení transakce

[2] zavedeno standardem SQL:2003

- *COMMIT* – potvrzení transakce (úspěšné provedení)
- *ROLLBACK* – zrušení transakce a návrat do původního stavu

Jazyk SQL má v současné době šest verzí standardu, které jsou vždy označovány dle roku vzniku:

- **SQL-89** – první specifikace standardu
- **SQL-92** – přinesl modifikace SQL schémat, zavedení tabulek s metadaty[3], vnějších spojení, kaskádního mazání/aktualizace podle cizích klíčů, množinové operace, transakce, kurzory, výjimky, ...
- **SQL-99** – rozšíření o regulární výrazy, rekurzivní dotazy, triggery, neskalární typy, objektové vlastnosti, ...
- **SQL:2003** – přináší XML rozšíření, standardizované sekvence a sloupce s automaticky generovanými hodnotami, odstranění datového typu *BIT*
- **SQL:2006** - definuje způsoby, jak importovat a ukládat XML data v databázích jak s nimi manipulovat uvnitř databáze a také jak publikovat data z databáze do XML formátu
- **SQL:2008** - upravuje některá předchozí rozšíření

V dnešní době je většina DBMS založena na standardu SQL-92 a SQL-99 (existují však rozdíly v implementaci[4]). Implementace významných částí standardů SQL-2006 a 2008 je pouze u některých produktů (například Oracle). Bohužel informace o tom, co která databáze ze standardů podporuje, jsou na internetu velmi špatně dostupné.

Celkový problém SQL spočívá v tom, že každá databáze do něj přidává své vlastní prvky. Většina databází navíc implementuje pouze části standardů, většinou dle aktuálních potřeb uživatelů (nebo obecně trhu), a tak je přenositelnost aplikací složitější. Jako příklad lze uvést vkládání více řádků pomocí jedno-

[3] metadata popisují strukturu samotné databáze (data o datech)

[4] informace o rozdílech implementace SQL v různých databázích lze nalézt na stránce http://troels.arvin.dk/db/rdbms/

ho `INSERT` dotazu. Dle standardu by podoba takového dotazu vypadala následovně (podpora v různých databázích je v tabulce č. 1.):

```
INSERT INTO tabulka VALUES (0,'Petr'), (1,'Pavel'), (2,'Mirek');
```

	MySQL 5	PostgreSQL 8.3	Firebird 2.1	Oracle 11g	MSSQL 2008
Podpora	ano	ano	ne	ne	ne

Tabulka č. 1 - podpora vkládání více řádků pomocí dotazu INSERT

2.2.4. Datové typy

Obecně se určuje typ dat pro sloupce tabulek. Kromě typu dat je někdy nutné uvádět i maximální možnou délku, u číselných údajů pak i počet desetinných míst. Obecně je lze rozdělit na:

- **číselné typy**
- **znakové typy**
- **typy pro datum a čas**
- **binární datové typy**
- **uživatelem definované typy**

Problémem dnešních databází jsou i datové typy. Každá databáze má podporu jiných typů, které jsou definovány standardy SQL. Některé mají dokonce své vlastní typy, jako například PostgreSQL, který má speciální typy pro ukládání IP[5] adres (`CIDR` a `INET`). Pro ukázku odlišností, jsou v tabulce č. 2 uvedeny některé datové typy, včetně jejich popisu, podle standardů SQL a v tabulce č. 3 je ukázáno, jak jsou skutečně implementovány v různých databázích.

DATOVÝ TYP	POPIS
Boolean	pravda, nepravda

[5] číslo, které jednoznačně identifikuje síťových zařízení (rozhraní)

Integer	celé číslo v rozmezí od -2^32 do +2^32 (2,147,483,648)
Tinyint	celé číslo v rozmezí od -128 do +127
Decimal	číslo s pevnou desetinou čárkou v rozmezí od -10^38+1 do 10^38+1
Real	číslo s pohyblivou desetinou čárkou (pro vysokou přesnost, například ve vědě) 1E+37 to +1E+37
Date	datum
Time	čas

Tabulka č. 2 – ukázka některých datových typů ve standardech SQL

V tabulce č. 3 jsou tučně vyznačeny datové typy, které nahrazují datové typy definovaných standardy. U nich může být rozdíl v maximálním povoleném rozsahu, ale také ve struktuře. Jak si lze povšimnout, většina databází nemá podporu typu `BOOLEAN`, ten je většinou simulován typem `CHAR(1)`.

DATOVÝ TYP	MySQL 5	PostgreSQL 8.3	Firebird 2.1	Oracle 11g	MSSQL 2008
Boolean	**tinyint(1)**	bool	**char(1), smallint**	**char(1)**	**bit, char(1)**
Integer	int	int4	integer	int	int
Tinyint	tinyint	int2	**smallint**	**smallint**	tinyint
Decimal	decimal (n1,n2)	decimal (n1,n2)	decimal (n1,n2)	decimal (n1,n2)	decimal (n1,n2)
Real	real	float8	**double**	real	real
Date	date	date	date	date	**datetime**
Time	time	time	time	date	**datetime**

Tabulka č. 3 – datové typy v různých databázích

2.2.5. Indexy

Index je pomocná datová struktura, která určuje pozici dat v tabulce na základě jejich hodnoty. Obecně můžeme říci, že slouží ke zrychlení přístupu k datům. Používat by se tak měly u všech sloupců, podle kterých se vyhledává, třídí nebo podle kterých se spojují tabulky.

Při vkládání dat do tabulek totiž nedochází k třídění jednotlivých záznamů a ukládají se většinou za sebe tak, jak byly postupně vloženy. Pokud budeme chtít z této tabulky později vybrat nějaká data, dle zadaného kritéria, budou se procházet všechny záznam a vybírat ty, které odpovídají. Procházení všech záznamů zamezují právě indexy, ve kterých jsou data organizována tak, aby bylo možné rychle vybrat pouze relevantní záznamy.

Typy indexů

Za základní indexy se považují ty, jež jsou dostupné v každé databázi (implementovány jsou jako B-stromy). Jedná se o indexy:

- **běžné** - pro zrychlení operací třídění a vyhledávání
- **unikátní** - pro zajištění unikátnosti hodnot ve sloupci (výjimku tvoří hodnota `NULL`)
- **primární** - pro jednoznačnou identifikaci řádků (nesmí obsahovat hodnoty `NULL`)

Vícesloupcové indexy

Index nemusíme mít jen na jednom sloupci v tabulce. V tabulkách se běžně pracuje s indexy, které jsou definovány nad více sloupci tzv. vícesloupcové indexy. U nich dochází k setřídění dat podle sloupců, nad kterými je index definován.

Více indexů nad jedním sloupcem

Samozřejmostí je možnost definování více indexů nad jedním sloupcem, kdy pro nalezení řádků se obvykle vybere ten, který množinu řádků více zredukuje. V praxi to znamená, že když budeme mít zadaná dvě kritéria výběru, například X = 2 a Y = 3, použije se jenom jeden index nad sloupcem X a řádky, které vyhovují druhé části podmínky, se musí dohledat záznam po záznamu.

Vhodná implementace indexů

V případě dotazů `INSERT`, `UPDATE` a `DELETE` obecně platí, že indexy tyto operace zpomalují. Správa indexů stojí určitou režii (například při každém vložení nebo odstranění záznamu se musí data znovu setřídit), proto nejsou indexy vhodné u tabulek, do kterých se především vkládá a minimálně se z nich čte.

2.2.6. Transakce

Jedná se o sérii příkazů čtení a zápisu do databáze. Hlavním účelem transakcí je zajištění uživatelům viditelnost konzistentního stavu databáze bez ohledu

na to, že více uživatelů přistupuje asynchronně ke stejným údajům a také aby selhání klienta nemohlo v žádném případě způsobit uvedení databáze do nekonzistentního stavu. V případě selhání databázového serveru již toto neplatí. Typickým příkladem transakce je převod peněz z jednoho konta na druhé. Transakce musí splňovat ACID:

- **Atomicita** - všechny operace s databází prováděné v rámci transakce jsou chápány jako jediná a nedělitelná operace
- **Konzistence** - V průběhu transakce může být databáze v nekonzistentním stavu, ale po úspěšném ukončení transakce je stav databáze považován za konzistentní.
- **Izolovanost** - souběžné provádění více transakcí nesmí ovlivňovat výsledek jednotlivých transakcí (každá transakce je zcela izolována od operací prováděných jinými transakcemi
- **Trvalost** – po úspěšném ukončení transakce jsou všechny změny provedené transakcí trvale uloženy

Pokud dojde k chybě nebo zrušení transakce v průběhu jakékoliv operace, pak musí dojít k obnově původního stavu databáze, který byl před zahájením transakce. Při úspěšném provedení transakce dojde k jejímu potvrzení (operace `COMMIT`). Opakem je operace `ROLLBACK`, jež může být vyvolána jak v důsledku neúspěšného provedení transakce, tak i zavoláním od klienta, většinou při odhalení nějakých nesrovnalostí v datech při jejich zpracování.

2.2.7. Trigger (spoušť)

Trigger je procedura, která je automaticky spuštěna DBMS jako reakce na specifikovanou akci v databázi. Používají se tam, kde je třeba při provedení nějaké akce provést i akce přidružené, například kvůli udržení konzistence dat v databázi. Spoušť je popsána třemi body:

- **Event (událost)** - změna v databázi, která vyvolá spuštění (příkazy `INSERT`, `UPDATE` a `DELETE`)

- **Condition (podmínka)** – dotaz nebo test, který je proveden, pokud je trigger aktivován
- **Action (akce)** – procedura, která je provedena při spuštění a pokud je splněna podmínka

Rozdělení spouští podle okamžiku vyvolání

- **BEFORE** - proveden před provedením změn v tabulce
- **AFTER** - proveden až po provedení změny v tabulce
- **INSTEAD OF** - proveden místo provedení změn v pohledu

2.2.8. Optimalizace při importu dat

Optimalizace se týkají vkládání (importu) většího počtu dat, které bude rychlejší a omezí se případné chyby, které by mohly nastat (typicky závislost mezi primárním a cizím klíčem). Obecně se jedná o následující kroky, které se provedou před vkládáním dat:

- odstranění primárních, cizích klíčů
- deaktivování (odstranění) indexů
- deaktivování (odstranění) trigerrů
- při použití transakcí vypnout tzv. autocommit (mód, kdy je okamžitě každý SQL příkaz implicitně potvrzen jako commit)
- zvýšení pracovní paměti na databázovém serveru

Po úspěšném vložení dat dojde ke znovu vytvoření primárních a cizích klíčů, aktivování nebo vytvoření indexů a triggerů a k upravení pracovní paměti na databázovém serveru, pokud byla předtím hodnota změněna.

3. ANALÝZA, NÁVRH ŘEŠENÍ A VÝBĚR TECHNOLOGIÍ

3.1. SOUČASNÝ STAV

Na internetu jsem se snažil najít aplikaci podobných funkcí. Ta nebyla naleznuta, proto má práce jistě přinese užitek širšímu okolí. Objevil jsem jen jednoduché PHP třídy umožňující import a export dat v různých formátech, knihovny a abstraktní databázové vrstvy nabízející jednotné rozhraní pro komunikaci s databázemi. Mezi ty nejvhodnější, a většinou také nejrozšířenější, patří:

- **Dibi** (http://dibiphp.com/cs/) – abstraktní databázová vrstva pro PHP
- **ADOdb** (http://adodb.sourceforge.net/) – abstraktní databázová vrstva pro PHP a Python
- **PDO** (http://cz.php.net/pdo) – PHP knihovna napsaná v jazyce C implementující společné rozhraní pro databáze
- **MDB2** (http://pear.php.net/package/MDB2) – balíček z repositáře PEAR nabízející abstraktní databázovou vrstvu pro PHP
- **ODBC** (http://cz.php.net/manual/en/book.uodbc.php) – API pro přístup k datům
- **PHP-ExcelReader** (http://sourceforge.net/projects/phpexcelreader/) – PHP knihovna umožňující práci s XLS soubory
- **PHPExcel** (http://www.codeplex.com/PHPExcel) – PHP knihovna umožňující práci s XLS soubory

3.2. NÁVRH ŘEŠENÍ

3.2.1. Struktura systému

Struktura systému by měla být navržena tak aby splnila dva důležité body **jednoduchost** a **přehlednost**. Při jejich dodržení bude následný vývoj a rozvoj mnohem jednoduší a rychlejší.

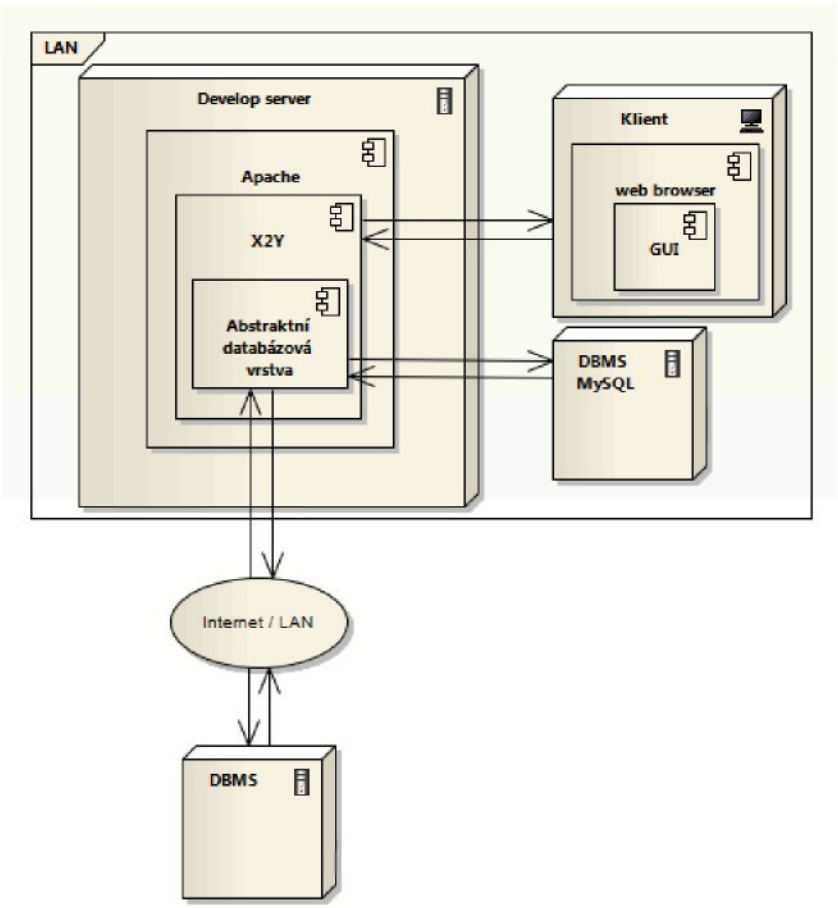
Je nutné vzít v úvahu potřebu práce s daty uložených v databázích. Obecně nelze říci kolik bude v databázi uloženo záznamů, těch může být i několik milionů, proto bude třeba dbát na povahu prováděných dotazů a dle toho systém následně optimalizovat, jinak by mohlo docházet, k překročení maximální povolené doby běhu skriptu nebo k nedostatku paměti. Z těchto důvodů by aplikace měla být kompaktní, jednoduchá, rychlá a implementující pouze potřebné funkce tak, aby zbytečně neplýtvala pamětí a prováděné operace byly prováděny relativně rychle (import milionu dat může trvat i několik hodin).

Systém by měl uživateli srozumitelně a přehledně sdělovat potřebné informace, zobrazovat chyby a nabízet nástroje pro práci s databázemi a daty.

Systém ponese název *X2Y* a jeho hierarchie bude rozdělena do tří vrstev:

1. **jádro** – stará se o základní funkce aplikace (přihlašování a odhlašování uživatele, session, inicializace připojení k databázím, ...)
2. **moduly** – jednotlivé části (soubory) aplikace zpřístupňující nástroje pro práci s databázemi a jejich daty
3. **abstraktní databázová vrstva** – nabízí unifikovaný přístup k databázím různého typu

Ve firmě CLWEB poběží aplikace na „develop“ serveru (vývojový server), kde nebude DBMS. Ten je umístěn na jiném počítači, přičemž připojování k němu bude probíhat v rámci sítě LAN. Aplikace nebude dostupná z internetu, ovšem sama o sobě k němu přístup mít bude.



obrázek č. 3 - diagram nasazení

Výše popsané a zobrazené nasazení aplikace není striktně dané. Jednou z dalších možností je provozování aplikace na samotném klientovi tzv. *localhost* (odkaz na právě používaný počítač), odkud by přistupovala k databázím prostřednictvím internetu nebo sítě LAN.

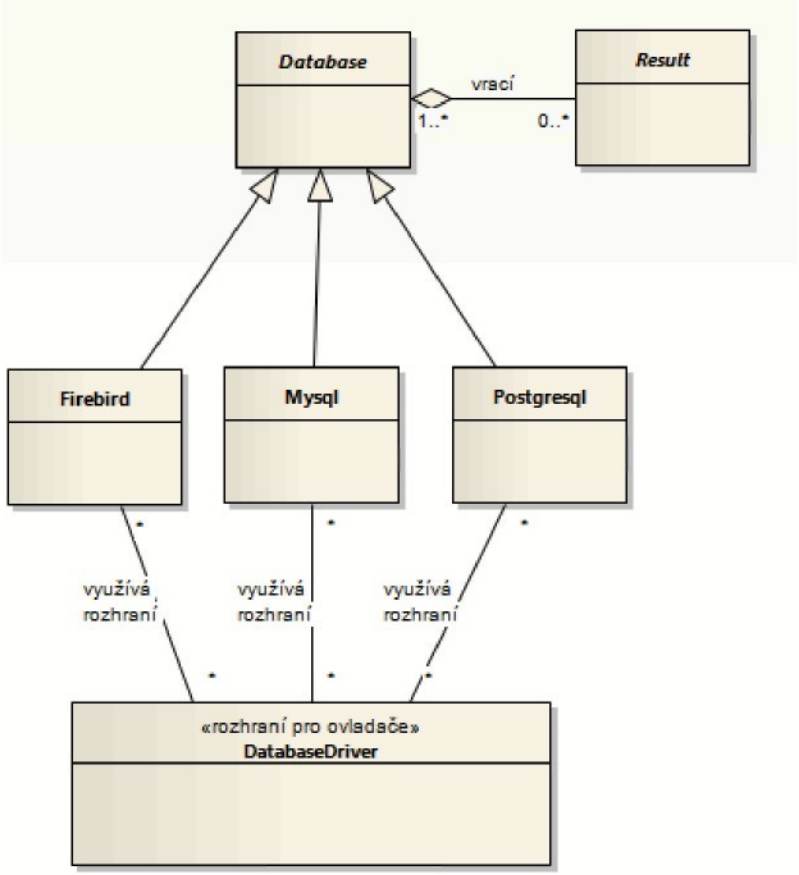
3.2.2. Abstraktní databázová vrstva

Tato vrstva poslouží k unifikovanému přístupu k databázím různého typu. Důvod návrhu vlastní vrstvy spočívá z několika nevýhod, které dostupné třídy a knihovny mají:

- většinou nemají metody pro získání informací o tabulkách a jejich sloupcích, indexech, ...
- některé neuvozují identifikátory (všechna klíčová slova SQL jsou přitom běžně užívaná anglická slova)
- nemají metody pro import a export dat
- neumožňují plnou kontrolu nad vrstvou (tu získáme vlastním návrhem a realizací)
- mnohdy složitá úprava funkčnosti dle potřeb aplikace

Navržená vrstva bude tyto body eliminovat (stanou se jejími přednostmi). Celkový koncept, je ukázán na obrázku č. 2, kde se nachází UML diagram tříd. Základními prvky budou:

- abstraktní třída `Database`
- třída `Result`
- rozhraní `DatabaseDriver` (implementuje potomek `Database`)
- ovladače pro konkrétní typ DBMS (potomci odvození od tříd `Database` a `Result`)



obrázek č. 4 - diagram abstraktní databázové vrstvy

Abstraktní třída `Database` a třída `Result` implementují metody, které budou společné pro všechny typy databází. První z nich bude obsahovat obecné metody pro práci s databází (vkládání, upravování, výběr a mazání záznamů z databází, export a import dat, ...), oproti tomu `Result` bude mít metody pro práci s výsledky (počet vrácených a ovlivněných záznamů, získání jednoho záznamu, uvolnění použitých prostředků, ...). Jelikož bude třída `Database` abstraktní, nebude možné vytvářet její instance. Tato možnost je přenechána až na její potomky (ovladače).

Ovladač bude jeden soubor, jenž obsahuje dva potomky přidávající metody, které musí být pro každý typ databáze implementovány jinak. Ty budou definované rozhraním `DatabaseDriver`. Ovladače budou muset splňovat následující pravidla:

1) ovladač musí být jeden soubor
2) název souboru s ovladačem musí být typu `Typdatabaze.class.php` (tzn. Firebird.class.php, Mysql.class.php, Postgresql.class.php, ...)
3) název potomka třídy `Database` musí být typ databáze začínající velkým písmenem, tzn. Firebird, Mysql, Postgresql, ...

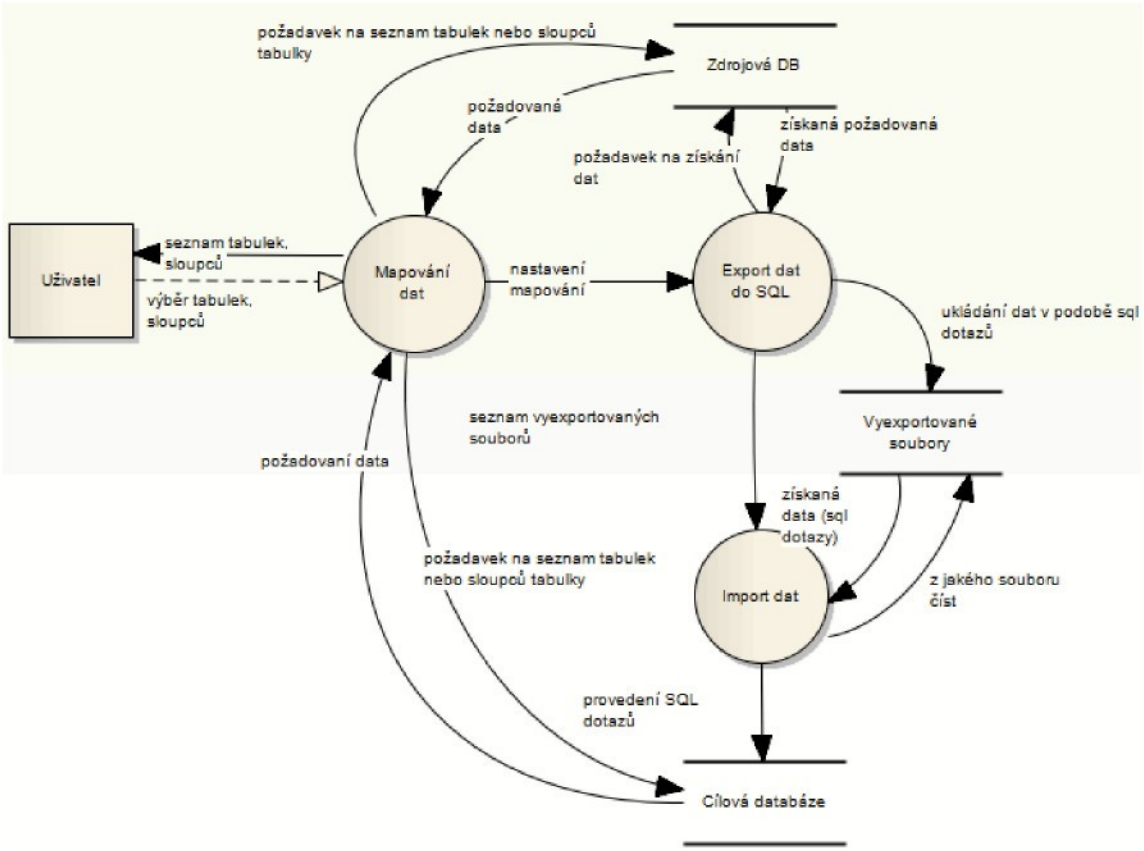
V praxi to bude fungovat tak, že pokud budeme chtít pracovat s databází typu Firebird, vytvoříme instanci objektu `Firebird`, čímž zároveň vytvoříme spojení. Provedeme-li poté na databázi jakýkoliv dotaz, dojde k vytvoření instance třídy `Result`. S vráceným výsledkem dotazu pak budeme pracovat jako s objektem. V případě provedení více dotazů, budeme mít možnost pracovat se všemi bez sebemenších problémů, díky zapouzdření dat do jednotlivých objektů.

Návrh předpokládá snadnou rozšiřitelnost o další typy pouhým přidáním ovladače a základní podporu databází MySQL, Firebird a PostgreSQL.

3.2.3. Přenos dat mezi databázemi

Problematiku samotného přenosu dat jsem nejdříve konzultoval s firmou CLWEB. Dospěli jsme k názoru, že nejčastěji je potřeba namapovat data ze sloupců jedné tabulky (ve zdrojové databázi) do tabulky jiné (v cílové databázi). Samotný přenos dat bude probíhat do *„čistých"* tabulek tzn. tabulek, kde nebudou klíče, indexy a kde budou vypnuté triggery. Kopírovaná data budou tzv. *„surová"*, tzn. že se kopírovat nebudou definice klíčů, indexů, triggerů atd.. Tyto věci jsou vždy specifické v rámci aplikace, pro kterou se data kopírují.

Celý koncept přenosu dat je znázorněn na obrázku č. 5 (DFD diagram).

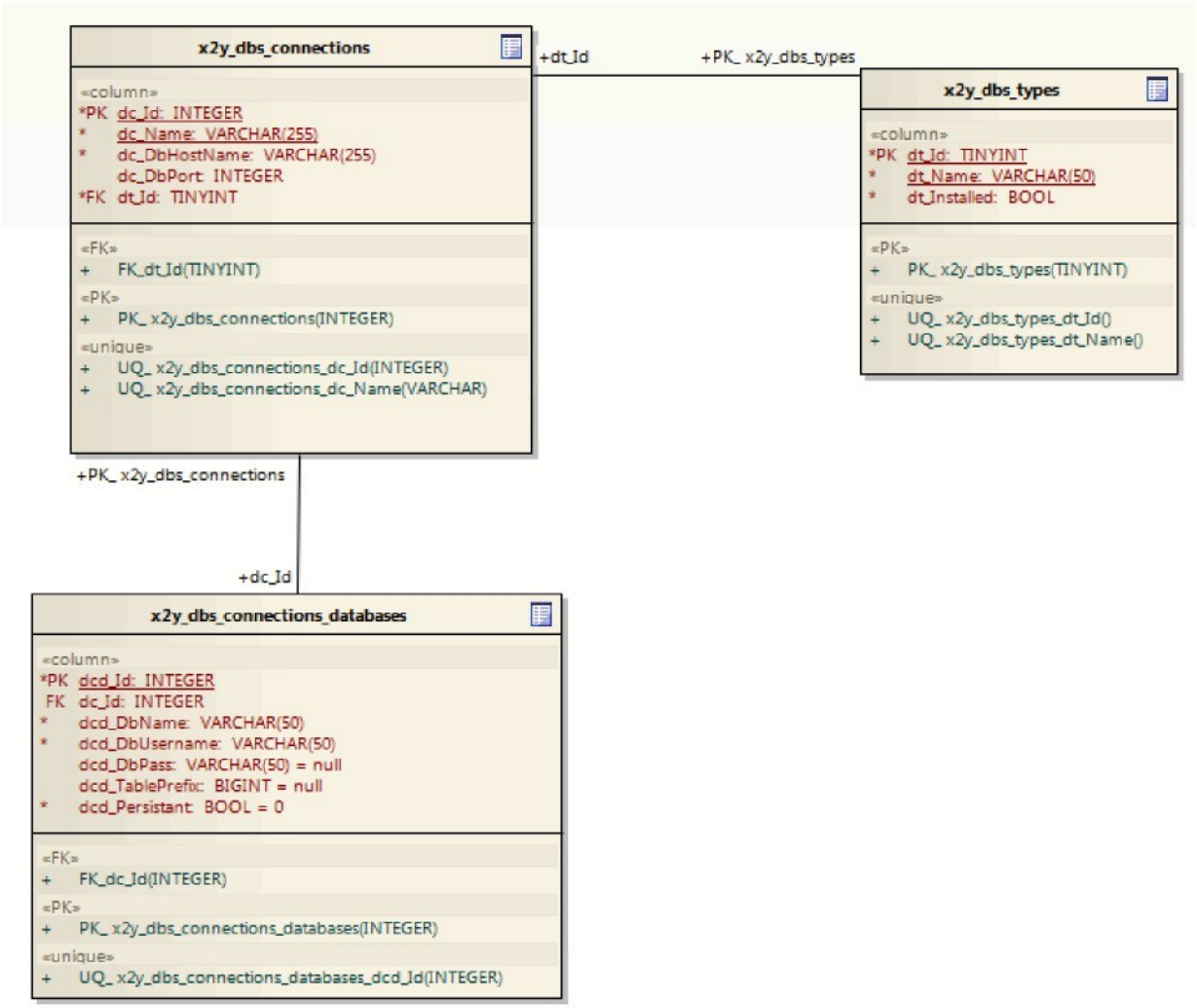


obrázek č. 5 - diagram toku dat

Nejdříve uživatel namapuje data ze zdrojové databáze do cílové. Celý proces kopírování bude založen na souborech typu SQL, do kterých se namapovaná data vyexportují jako SQL příkazy odpovídající dialektu cílové databáze. Následně pak dojde k postupnému přečtení všech souborů a vykonání příkazů v nich uložených. Záměrně zde existuje mezičlánek, kterým jsou zmiňované vyexportované SQL soubory. Ten nám umožní využít tyto soubory jako zálohu dat, což se může v mnoha případech hodit.

3.2.4. Struktura databázových tabulek

Je složena ze tří tabulek, které mají prefix `x2y_`. Ten mají tabulky z důvodu jejich případného použití v databázi, kde již jsou jiné tabulky. Díky němu pak nemůže dojít k jejich případné záměně a tabulky budou dobře odlišitelné od ostatních. Jejich strukturu včetně závislostí, lze vidět níže na obrázku č. 3, kde se nachází ERD diagram.



obrázek č. 6 - diagram struktury databázových tabulek

Z diagramu je patrné, že se jedná o jednoduchou strukturu. Ta bude sloužit především k uchování informací o připojeních, aby je nebylo nutné zadávat vždy ručně. Konkrétnější popis jednotlivých tabulek, včetně informací co budou uchovávat:

- **x2y_dbs_connections** - slouží k uchování základních informací o připojení (název připojení, jméno hostitele, port, typ DBMS)
- **x2y_dbs_connections_databases** - slouží k uchování nastavení pro připojení k databázím (název databáze, uživatelské jméno a heslo, prefix tabulek, persistentní spojení) pro konkrétního hostitele
- **x2y_dbs_types** - slouží k uchování informací o dostupných ovladačích, zda jsou nainstalované (lze je použít) popřípadě odinstalované (nelze je použít)

3.3. VÝBĚR TECHNOLOGIÍ

3.3.1. XHTML

XHTML je v dnešní době nejvýznamnější značkovací jazyk pro tvorbu webových stránek vyvinutý W3C. Jedná se o značkovací jazyk. To znamená, že se v něm používají speciální značky nazvané tagy, které určují význam textu v dokumentu. Dokument se řídí definicí podle souboru DTD.

3.3.2. CSS

Je technologie kaskádových stylů, která nám umožňuje oddělit vzhled a obsah HTML stránek. Díky této vlastnosti se dá v budoucnu snadno měnit vzhled HTML stránek, bez zásahu do jejich obsahu, zároveň nám zpřehledňuje kód HTML stránek.

3.3.3. JavaScript

JavaScript je multiplatformní, objektově orientovaný a v současné době nejrozšířenější skriptovací jazyk, který se používá v internetových stránkách. Jedná se o klientský skript (běží na straně klienta) což má za následek jistá bezpečností omezení, např. nelze pracovat se soubory na disku.

3.3.4. PHP

Jedná se o interpretovaný skriptovací jazyk, který je určen především pro programování dynamických stránek. PHP umožňuje procedurální i objektově orientovaný přístup k programování. Patří také mezi jazyky, kde není nutné předem definovat typ proměnných, navíc jakákoli proměnná může kdykoli změnit svůj typ. Jazyk je šířen jako open-source, může být tedy modifikován podle potřeb, stejně tak existuje i mnoho podpůrných knihoven a nástrojů, které jsou pro něj dostupné. Tento jazyk nemá speciální požadavky na uživatelův počítač, neboť PHP skripty se provádějí na serveru a klientovy se posílají již jen obyčejné HTML stránky. Díky tomu je zajištěna kompatibilita s různými systémy.

3.3.5. AJAX

AJAX je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich opakovaného načítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů. Tyto aplikace využívají následujících technologií:

- (X)HTML a CSS pro prezentaci informací
- DOM a JavaScript pro zobrazování a dynamické změny informací
- aplikační rozhraní XMLHttpRequest pro (a)synchronní výměnu dat s webovým serverem obvykle je používán formát XML, ale je možné užít jiný formát včetně HTML či prostého textu)

3.3.6. MySQL

MySQL je multiplatformní relační databázový systém, který je k dispozici jak pod bezplatnou licencí GPL, tak i pod komerční verzí licence. Velmi často bývá používána kombinace MySQL, PHP a Apache jako základní software webového serveru. Práce s tímto systémem se dá využít v C, C++, Java, Perl, PHP, Python, Tcl, Visual Basic nebo .NET. MySQL přináší především stabilitu, rychlost (hlavně pro čtení), výkonnost a hardwarovou nenáročnost.

3.3.7. PostgreSQL

PostgreSQL je multiplatformní relační databázový systém, který je šířen jako open-source. Má za sebou více než 15 let aktivního vývoje a má vynikající pověst pro svou spolehlivost a bezpečnost. PostgreSQL nabízí plnou podporu ACID transakcí, plně podporuje cizí klíče, pohledy, spouště a uložené procedury. Pro přístup k této databázi existují rozhraní pro JDBC, ODBC, dbExpress, Open Office, PHP, .NET a PERL.

3.3.8. Firebird (FirebirdSQL)

Firebird je multiplatformní relační databázový systém, který je šířen jako open-source. Technologie Firebirdu je používána již 20 let, je totiž založený na zdrojových kódech systému InterBase 6.0 uvolněných firmou Borland Software Corp. To z něj dělá velmi zralý a stabilní produkt, který nabízí plnou podporu ACID transakcí, referenční integritu, multigenerační architekturu, velmi nízké hardwarové nároky, excelentní propustnost při víceuživatelském zpracování, vysoký výkon, mocnou podporu pro uložené procedury a spouště a velké množství způsobů jak přistupovat k databázi (nativní/API, ODBC, OLEDB, .NET provider, JDBC, Python modul, PHP, Perl atd.).

3.4. POUŽITÍ DOSTUPNÝCH PHP SKRIPTŮ

Aplikace bude využívat následujících PHP skriptů, které jsou volně dostupné na internetu:

Form

- třída pro tvorbu webových formulářů
- umožňuje jednoduchou a rychlou tvorbu formulářů v PHP bez nutnosti psaní XHTML kódu
- je šířena pod GNU licencí
- http://www.dogdoo.net/php/

```php
<?php
include_once('class.form.php');
$form = new genForm;
$form->startForm(basename($_SERVER['PHP_SELF']));
   $form->textInput('email','Email Address',false,'block');
   $form->textareaInput('comments','Enter Your Comments',false,'block');
   $form->insertBR();
   $form->submitButton();
$form->closeForm();

if(!$output = $form->getForm()) { die("error: " . $form->error); }
else { echo $output; }
?>
```

obrázek č. 7 - ukázka práce s třídou Form

Htmltmpl

- šablonovací systém
- vychází z šablonovacího systému HTMLTMPL od Tomáše Stýbla (http://htmltmpl.sourceforge.net/) pro PHP a Python
- upravená verze od Jakuba Vrány (http://php.vrana.cz/sablony.php)
- používá značky ve tvaru `<TMPL_VAR>` nebo `<!-- TMPL_VAR -->` z původního systému HTMLTPL
- kompiluje TMPL šablony do PHP kódu
- do cache ukládá přímo nativní PHP kód
- šířeno pod GNU licencí

Třída *Form* bude v aplikaci použita k vytvoření několika dynamických formulářů či jejich částí. *Htmltmpl* bude ve výsledné aplikaci oddělovat aplikační vrstvu aplikace od prezentační.

4. REALIZACE

4.1. ADRESÁŘOVÁ STRUKTURA

Základní adresářová struktura aplikace je následující:

- ***cache/*** - adresář pro ukládání zkompilovaných šablon
- ***css/*** - adresář s CSS stylem, využitý v aplikaci k utvoření vzhledu, pozicování, ...
- ***exports/*** - adresář pro ukládání vyexportovaných souborů
- ***images/*** - adresář s obrázky, které aplikace využívá
- ***imports/*** - adresář pro uložení souborů nahraných na server za účelem importu dat
- ***include/ajax/*** - adresář s PHP skripty používaných pro AJAX
- ***include/class/*** - adresář obsahující třídy
- ***include/constants/*** - adresář se souborem obsahující konstanty
- ***include/functions/*** - adresář se soubory obsahující funkce
- ***include/pages/*** - adresář pro jednotlivé části aplikace
- ***include/templates/*** - adresář s nezkompilovanými šablonami použitých aplikací
- ***js/*** - adresář se souborem obsahující JavaScript
- ***logs/*** - adresář pro ukládání logů aplikace

4.2. SOUBOROVÁ STRUKTURA APLIKACE

Souborovou strukturu lze rozdělit na dvě části. První částí jsou soubory nutné pro běh aplikace (jádro), druhou částí jsou soubory, které jsou načítány jako moduly. Tyto moduly zpřístupňují veškeré funkce aplikace uživateli.

Hlavním souborem, který je vždy vykonán jako první, je `index.php`. Stará se o načítání všech potřebných souborů a bezpečnost. Načítá také další potřebné soubory a nachází se v kořenovém adresáři aplikace.

Další nezbytné soubory pro běh aplikace, včetně jejich umístění a popisu:

- ***index.php*** – jádro aplikace, načítá ostatní potřebné soubory, vytváří spojení s DB, ...
- ***config.php*** – konfigurační soubor aplikace (hlavní nastavení aplikace)
- ***main.inc.php*** - stará se o načítání stránek, dle požadavků a zobrazení výsledku běhu aplikace
- ***css/main.css*** – soubor s CSS stylem k utvoření vzhledu, pozicování, ...
- ***include/auth.inc.php*** – soubor umožňující přihlášení a odhlášení uživatele
- ***include/session.inc.php*** – soubor pro správu session
- ***include/constants/constants.inc.php*** – soubor obsahující použité konstanty
- ***include/functions/main_fn.inc.php*** – soubor obsahující systémové funkce (`__autoload`, `error_handler`, ...)
- ***include/class/Database.class.php*** – soubor s rozhraním pro ovladače a s abstraktní třídou zpřístupňující metody společné pro všechny typy databází
- ***include/class/db/Firebird.class.php*** – ovladač pro databáze Firebird
- ***include/class/db/Mysql.class.php*** – ovladač pro databáze MySQL
- ***include/class/db/Postgresql.class.php*** – ovladač pro databáze PostgreSQL

- ***include/functions/Messages.class.php*** – soubor pro zobrazování zpráv a informací uživateli
- ***include/class/Templates.class.php*** – soubor pro práci s šablonami

4.3. POPIS SOUBORŮ S PHP SKRIPTY

Všechny vytvořené soubory s PHP skripty obsahují hlavičku, ve které je uveden autor a popis funkcionality daného souboru. Samozřejmostí jsou komentáře u metod, funkcí, atributů a všech netriviálních operací. Níže je uvedena ukázka komentáře u funkce `error_handler` včetně popisu definice jednotlivých argumentů a návratové hodnoty.

```
/**
 * Funkce elegantne zobrazujici chyby pomoci abstraktni tridy Messages
 *
 * @param integer $errno    -cislo chyby
 * @param string  $errstr   -zprava chyby
 * @param string  $errfile  -soubor, kde doslo k chybe
 * @param integer $errline  -radek, kde doslo k chybe
 * @return bool
 */
```

obrázek č. 8 - ukázka komentáře u funkce `error_handler`

4.4. ZABEZPEČENÍ SOUBORŮ S PHP SKRIPTY

Veškeré soubory s PHP skripty, kromě souborů s funkcemi, třídami, skriptů pro AJAX a konfiguračního souboru (tyto soubory samy o sobě nic nevykonávají), je třeba zabezpečit proti vykonání jejich kódu, pokud by byly volány přímo (nevyžádala by si je aplikace).

Zabezpečit se dají jednoduše tak, že je budeme považovat za moduly aplikace, které mohou být načítány souborem `index.php` nebo *main.inc.php*. Při spuštění aplikace je definována konstanta `LOADED_AS_MODULE = 1`, jejíž existence modulům říká, že jsou právě načteny souborem aplikace. Všechny

stránky, které toto zabezpečení mají, obsahují krátký kód, kde dochází k ověření, zda je konstanta definována a pokud tomu tak není, okamžitě dojde k zastavení běhu aplikace.

4.5. LADICÍ MÓD

Pokud je definována konstanta `X2Y_DEBUG = 1` běží aplikace v tzv. „debug módu" (ladicí mód). Tento mód je v aplikaci z důvodu jejího testování. Po jeho zapnutí se ve spodní části, pod patičkou stránek, zobrazí *„DEBUG WINDOW"* (ladicí okno), kde jsou k vidění základní systémové informace, vykonané SQL dotazy na databáze a případně chyby, které generovalo samotné PHP. Okno se dá schovat po kliknutí na odkaz `DEBUG WINDOW`.

obrázek č. 9 - ukázka zapnutého debug módu

4.6. ZOBRAZOVÁNÍ CHYBOVÝCH STAVŮ A ZPRÁV

O zobrazování chybových stavů a zpráv se stará abstraktní třída `Messages`. Tato třída je využívána jak pro výpis chybových stavů a zpráv aplikace, tak i chyb, upozornění a oznámení generovaných přímo samotným PHP. Pomocí funkce `set_error_handler("error_handler", E_ALL)` v souboru `index.php` dochází k přesměrování všech generovaných stavů na funkci, která pro každý z nich zavolá uvedenou třídu s konkrétními parametry.

Pokud běží aplikace v ladicím módu, zobrazují se kromě chybových hlášení aplikace také informace o tom, v jakém souboru chyba vznikla a na jakém řádku se nachází.

V aplikaci jsou rozlišeny čtyři druhy zpráv a dva druhy chybových stavů:

- **Informační zpráva** - Sdělování informací uživatelům. Příkladem může být informace, že byl nalezen nový ovladač.
- **Všeobecná zpráva** – Sdělování zpráv uživatelům, že operace proběhla úspěšně. Příkladem může být, že export dat proběhl v pořádku.
- **Varovná zpráva** – Sdělování zpráv uživatelům, že se operace nezdařila (většinou z důvodu špatného nastavení uživatelem). Informace, které se zobrazují například, pokud nedojde ke spojení s cílovou nebo zdrojovou databází.

- **Všeobecná chyba** – Chyby, které nastanou při běhu aplikace, ale nezpůsobí její ukončení. Příkladem může být chyba, která nastala při pokusu o import dat.
- **Kritická chyba** – Kritické chyby jsou ty, které okamžitě ukončí běh aplikace, aby nemohlo dojít k jejich zneužití a aby nevyvolali nestandardní běh aplikace. Jedná se především o nepředvídatelné chyby. Například při vykonávání běžného SQL dotazu může být databázový server nedostupný. Chyba nenastala díky uživateli, nebo aplikaci, ale její běh musí být zastaven, protože by mohlo dojít k nestandardnímu běhu, protože nedojde k získání požadovaných dat.

4.7. ŠABLONOVACÍ SYSTÉM

Základním cílem šablonovacího systému je oddělit aplikační vrstvu aplikace od prezentační. V praxi jde o to, abychom v programovacím jazyce negenerovali přímo HTML kód, ale vytvořili si šablonu pomocí tagů šablonovacího systému, která bude generovat (X)HTML. Programátor tak píše pouze v programovacím jazyce a nemusí se starat o design stránek, který má na starost kodér (X)HTML a CSS (webdesignér). Webdesignér pak obstará pouze šablonu, aniž by musel umět programovat. Požadovaná data jsou šabloně předávána typicky prostřednictvím proměnných.

Výhody, které šablonovací systémy přináší:

- oddělení aplikační a prezentační vrstvy
- vymezení rolí při vývoji a správě (programátor x kodér či webdesignér)
- zamezení duplicit v kódu
- snadná lokalizace
- nezávislost na použitém programovacím jazyce (šablonovací jazyk může být implementován v různých programovacích jazycích)

Šablonovacích systémů existuje mnoho např. Smarty, P.E.T. a jiné. Některé systémy jsou mocnými nástroji, ale jsou poměrně složité a velké. Taktéž mají svou syntaxi, kterou se člověk musí naučit.

Aplikace X2Y využívá upravený šablonovací systém HTMLTMPL od Jakuba Vrány. Výběr tohoto systému spočíval v jeho velikosti, jednoduchosti a nenáročnosti. Syntaxe je podobná jazyku HTML a tak je velice jednoduché se jí naučit. Využitelnost tohoto systému lze najít jak u menších, tak i větších projektů.

Samotný systém se skládá z několika mála funkcí pro práci se šablonami, proto byl tento systém upraven do podoby abstraktní třídy. Načtení konkrétní šablony v PHP lze provést následujícím kódem `Templates::LoadTemplate("NAZEV_SABLONY", PARAMETRY)`, kde parametry představují pole s proměnnými, které dané šabloně chceme předat.

4.7.1. Syntaxe šablon

Příkazy musí být uzavřeny v `<PRIKAZ>` a lze je zapisovat ve dvou podobách:

- `<TMPL_STATIC>`
- `<!-- TMPL_STATIC -->`

Jejich používání je velice jednoduché, jak lze vidět na následujícím obrázku, kde je použit příkaz `TMPL_VAR` na výpis proměnné `row_count`.

```
<div class="rows-count">
    <p>Vráceno záznamů: <TMPL_VAR row_count></p>
</div>
```

obrázek č. 10 - ukázka použití příkazu `TMPL_VAR`

Některé z příkazů mohou obsahovat parametry:

- **NAME** – název proměnné, souboru
- **ESCAPE** – jakým způsobem se mají data escapovat
- **GLOBAL** – určuje, zda se jedná o globální proměnnou

I parametry lze zapisovat dvěma způsoby a to z uvozovkami a nebo bez nich:

- `<TMPL_VAR nazev_promenne ESCAPE="HTML">`
- `<TMPL_VAR nazev_promenne ESCAPE=HTML>`

Existuje také možnost psát komentáře ve tvaru: `### nejaky komentar`. Vše, co se nachází za ### je odstraněno při kompilaci šablony.

4.7.2. Zabezpečení práce s větším množstvím dat

Jednou z předních vlastností aplikace je možnost pracovat i s větším množstvím dat, konkrétně s milióny záznamů. Této vlastnosti je dosaženo zamezením následujících dvou bodů:

- překročení maximální doby běhu skriptu
- vyčerpání paměti

4.7.3. Zamezení překročení maximální doby běhu skriptu

Většinou je na serverech nastaven maximální limit pro běh skriptu na 60 sekund. Toto je v mnoha případech pro aplikaci *X2Y* velké omezení, protože import a export jsou náročné operace z hlediska času. I když se dá tato hodnota změnit v `php.ini` nebo `funkcí set_time_limit()` nelze dopředu říci, jaká maximální hodnota bude dostačující. Pokud ovšem funkci `set_time_limit()` dáme jako parametr nulu, poběží skript po celou dobu jeho vykonávání.

4.7.4. Zamezení vyčerpání paměti

Pokud v PHP položíme dotaz na databázi, který vrací konkrétní data, tak dojde k uložení celého výsledku v paměti procesu[6]. V případě exportu miliónů záznamů z tabulky by mohlo dojít k vyčerpání paměti. Tento problém lze řešit jedním ze dvou následujících způsobů:

[6] nejedná se o přidělenou paměť PHP skriptu

1) Pokud je to možné, použít tzv. "*unbuffered*" dotazy, kdy se žádná data nikde neukládají (lze použít pouze pro databáze MySQL, SQ-Lite a Sybase).
2) Prvním dotazem zjistit, kolik bude vráceno záznamů. V cyklu pak číst jednotlivé záznamy, které se následně zpracují, a na konci dojde k uvolnění použitých zdrojů. Poté položit dotaz na další data. Takto pokračovat, dokud se vše nepřečte.

Aplikace také průběžně uvolňuje všechny nepotřebné prostředky (proměnné, objekty, ...), tak aby nedošlo k vyčerpání přidělené paměti PHP skriptu.

4.8. IMPORT A EXPORT DAT

Pro import a export dat měly být dostupné tři typy souborů, které se nejběžněji používají a to CSV, XML a XLS. Dalším vhodným typem je SQL, který byl použit i při kopírování dat.

Ke čtení a vytváření souborů typu XLS jsou volně dostupné třídy, napsané v PHP, *PHP Excel 2007* a *PHP-ExcelReader*. Ty ovšem nejdříve získají nebo přečtou veškerá data a teprve následně dojde k vytvoření souboru, či ke zpřístupnění přečtených dat. Tento způsob při větších souborech vede rychle k vyčerpání přidělené paměti. Samotné vytvoření skriptů by bylo časově náročné.

Při ukládání souborů v Excelu není problém zvolit formát CSV, který je plně dostačující a lze ho jednoduše zpracovat při následném importu. V tomto případě tak dochází k plnohodnotné náhradě za XLS. Pro export už ovšem dostačující není. Příčinou je omezení maximálního počtu řádků na jeden sešit. V *Excelu 2003* 65 536 řádků a v *Excelu 2007* 1 048 576 řádků. Nelze ani rozložit data do více sešitů, kdy by došlo k vyřešení problému.

Veškeré problémy řeší formát MHT. Používá se v souvislosti s webovými stránkami. Jedná se o webový archív, kde je jeden soubor s příponou MHT ob-

sahující vlastní XHTML kód, obrázky, CSS a další části webových stránek. Excel tento soubor používá k možnosti publikování dat na internetu. Umožňuje navíc definovat více sešitů, čímž odpadá omezení na maximální počet řádků, které *Excel* dokáže zobrazit v jednom sešitu.

X2Y tento formát souborů podporuje a je možné ho zvolit pro export dat. K tomuto účelu byl vytvořen vlastní algoritmus. Ukázka struktury souboru vytvořeného aplikací, je na obrázku č. 11.

```
MIME-Version: 1.0
X-Document-Type: Workbook
Content-Type: multipart/related; boundary="----=_NextPart_01C9897A.A5E19690"

------=_NextPart_01C9897A.A5E19690
Content-Location: file:///C:/268B22B2/set2.htm
Content-Transfer-Encoding: quoted-printable
Content-Type: text/html; charset="windows-1250"

<html xmlns:v=3D"urn:schemas-microsoft-com:vml"
xmlns:o=3D"urn:schemas-microsoft-com:office:office"
xmlns:x=3D"urn:schemas-microsoft-com:office:excel"
xmlns=3D"http://www.w3.org/TR/REC-html40">
<head>
  <!--[if gte mso 9]><xml>
    <x:ExcelWorkbook>
      <x:ExcelWorksheets>
                <x:ExcelWorksheet>
                  <x:Name>1</x:Name>
                  <x:WorksheetSource HRef=3D"set2_files/sheet1.htm"/>
                </x:ExcelWorksheet>
      </x:ExcelWorksheets>
    </x:ExcelWorkbook>
</xml><![endif]-->
</head>
<body>
</body>
</html>
------=_NextPart_01C9897A.A5E19690
Content-Location: file:///C:/268B22B2/set2_files/sheet1.htm
Content-Transfer-Encoding: quoted-printable
Content-Type: text/html; charset="windows-1250"

<html xmlns:o=3D"urn:schemas-microsoft-com:office:office"
xmlns:x=3D"urn:schemas-microsoft-com:office:excel"
xmlns=3D"http://www.w3.org/TR/REC-html40">
<head>
<meta http-equiv="Content-type" content="text/html;charset= charset=windows-1250" />
</head>
<body>
 <table>
   <tr>
   <td>13</td>
   <td>Firebird</td>
   <td>1</td>
   </tr>
   <tr>
   <td>14</td>
   <td>Mysql</td>
   <td>1</td>
   </tr>
   <tr>
   <td>15</td>
   <td>Postgresql</td>
   <td>1</td>
   </tr>
  </table>
</body>
</html>
------=_NextPart_01C9897A.A5E19690--
```

obrázek č. 11 – ukázka struktury souboru typu MHT

Jednotlivé části souboru jsou rozděleny řetězcem `------=_NextPart_01C9897A.A5E19690`. V první části je XML, kde je definice všech sešitů. Následují data v podobě XHTML tabulky pro jeden sešit. Pokud bychom měli sešitů více, nacházelo by se v souboru více takových částí.

4.9. ZPĚTNÁ KOMPATIBILITA

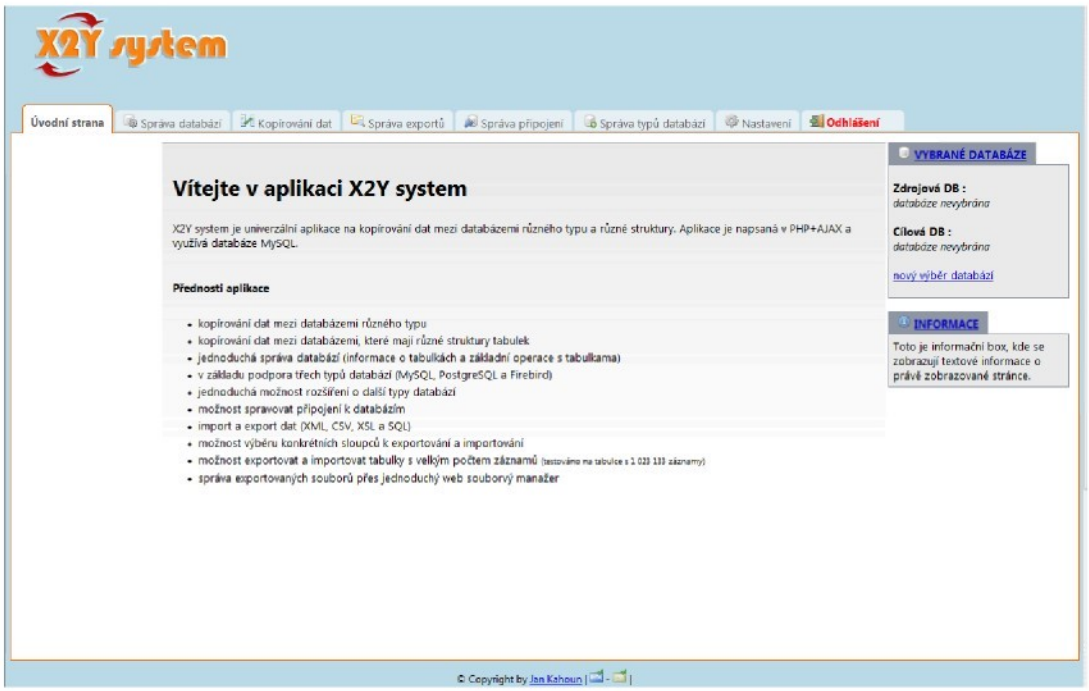
Aplikace je navržena tak, aby byla umožněna zpětná kompatibilita s nižšími verzemi DBMS, které jsou ze stran výrobců stále podporované (tabulka č. 4). Aby bylo možné toto splnit, jsou používány pouze SQL příkazy a vlastnosti, které dané databáze podporují i v nižších verzích.

MySQL	PostgreSQL	Firebird
5.1	8.3	2.1
5.0	8.2	2.0
4.1	8.1	1.5
	8.0	
	7.4	

Tabulka č. 4 – podporované verze databází

4.10. TVORBA GRAFICKÉHO NÁVRHU

Vzhled stránek měl být velice jednoduchý, přehledný a dodržovat standardy. Pro tvorbu tedy bylo použito XHTML 1.0 Strict a CSS stylů. Bylo také dbáno na validitu kódu a na správné zobrazení stránek v nejvíce používaných webo-



vých prohlížečích. Snažil jsem se dosáhnout přehlednosti a dobré čitelnosti.

obrázek č. 12 – náhled výsledného grafického vzhledu

4.11. VÝSLEDNÉ FUNKCE APLIKACE

Nejen že aplikace splňuje obecně definované cíle, ale také přináší další vlastnosti a funkcionality. Výsledné vlastnosti a funkce aplikace jsou následující:

- kopírování dat mezi databázemi různého typu
- kopírování dat mezi databázemi, které mají různé struktury tabulek
- jednoduchá možnost rozšíření o další typy databází pouhým přidáním ovladače pro konkrétní databázi
- podpora třech typů databází MySQL, PostgreSQL a Firebird
- import dat ze souborů ve formátech XML, CSV a SQL s možností namapování dat do příslušných sloupců v tabulce
- export dat do souborů ve formátech XML, CSV, MHT a SQL s možností výběru konkrétních sloupců, které budou exportovány
- možnost exportovat, importovat a kopírovat tabulky s velkým počtem záznamů (testováno na tabulce s 4 092 952 záznamy)
- možnost spravovat připojení k databázím
- možnost položit SQL dotaz na konkrétní databázi
- jednoduchá správa tabulek
- zobrazování informací o tabulkách (seznam sloupců a jejich vlastnosti, definované klíče a indexy)
- správa vyexportovaných souborů pomocí web souborového manažeru
- možnost zvolit si, v jakém kódování, z nejpoužívanějších kódových sad windows-1250, utf-8 a ISO 8859-2 podporujících češtinu, budou data exportována
- možnost komprimovat vyexportovaná data
- jednoduchá možnost nastavení aplikace přes webový formulář

5. TESTOVÁNÍ APLIKACE

5.1. POSTUP TESTOVÁNÍ

Jednotlivé části aplikace (PHP skripty) jsem vždy po vytvoření ladil pomocí zkušebních databázových struktur (tabulek) a dat, tak aby se chovaly správně. Po tomto odladění jsem, danou část implementoval do aplikace, kde jsem ji testoval s ostatními částmi aplikace. Také jsem testoval JavaScript přes doplněk Firebug v prohlížeči FireFox a vygenerovaný XHTML kód na validitu pomocí aplikace dostupné na http://validator.w3.org/, která kontroluje dodržování standardů (X)HTML.

5.2. PODPOROVANÉ PROHLÍŽEČE

Webová aplikace byla otestována a je plně funkční v následujících prohlížečích:

- **Opera verze 9.5**
- **Mozilla firefox verze 3.0.8**
- **Internet explorer verze 7 a 8**
- **Safari verze 4**

5.2.1. Ostatní webové prohlížeče

Prohlížeči se předává validní XHTML kód s JavaScriptem. Jelikož existují rozlišnosti v prohlížečích s interpretací jazyka JavaScript, je pravděpodobné, že by mohl být problém s funkčností stránek. Každý prohlížeč také jinak podporuje CSS, takže není zaručeno správné zobrazení grafické úpravy, což by mohlo mít vliv na funkčnost aplikace. Díky těmto okolnostem se používání nepodporovaných prohlížečů nedoporučuje.

5.3. OVĚŘENÍ FUNKČNOSTI APLIKACE

Funkčnost výsledné aplikace byla ověřena několikanásobným importem a exportem dat ve všech formátech s využitím všech podporovaných databází. Následně byla otestována funkčnost přenosu dat, mezi všemi typy databází. Kromě Firebirdu, kde jsem využil k softwaru přiložené ukázkové databáze *EMPLOYEE.fdb*, probíhalo testování na mnou vytvořených databázových strukturách s daty, která jsem si připravil. Veškeré mnou prováděné testy proběhly úspěšně na lokální počítačové sestavě *Intel Core 2 Duo 3.2 Ghz, 4 GB RAM 800 Mhz* s *Windows Vista SP 1*.

V poslední fázi byla testována časová náročnost na export a import dat. Průměrné naměřené hodnoty jsou uvedeny v následujících tabulkách č. 5 a 6. Tabulky s hodnotami jsou pouze orientační. Výsledek totiž ovlivňuje mnoho faktorů:

- výkonnost počítače(ů)
- použité verze softwaru (Apache, DBMS, ...)
- typ databáze (MySQL, Firebird a PostgreSQL)
- typ souboru s daty (SQL, CSV, XML a MHT)
- velikost dat
- struktura tabulek (rozdílné množství sloupců, různé datové typy, ...)
- zda v daný okamžik databáze nebo počítač zpracovává více požadavků nebo jen jeden

	2 000 záz.	23 000 záz.	210 000 záz.	4 100 000 záz.
CSV	0.3 s.	6 s	67 s	789 s
MHT	0.4 s	8 s	69 s	893 s
SQL	0.3 s	5 s	61 s	718 s

XML	0.4 s	8 s	70 s	876 s

Tabulka č. 5 – časové průběhy při exportu

	2 000 záz.	**23 000 záz.**	**210 000 záz.**	**4 100 000 záz.**
CSV	2 s	35 s	315 s	4 135 s
SQL	6 s	150 s	1 278 s	8 668 s
XML	15 s	356 s	2 835 s	16 720 s

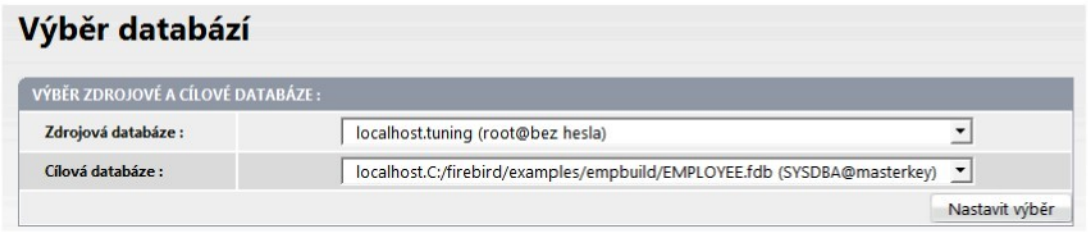
Tabulka č. 6 – časové průběhy při importu

6. UKÁZKA PŘEVODU DAT V APLIKACI X2Y

Převod dat probíhá ze zdrojové databáze do databáze cílové. V aplikaci je to realizováno ve čtyřech krocích:

1) Namapování dat
2) Kontrola namapování
3) Export dat
4) Import dat

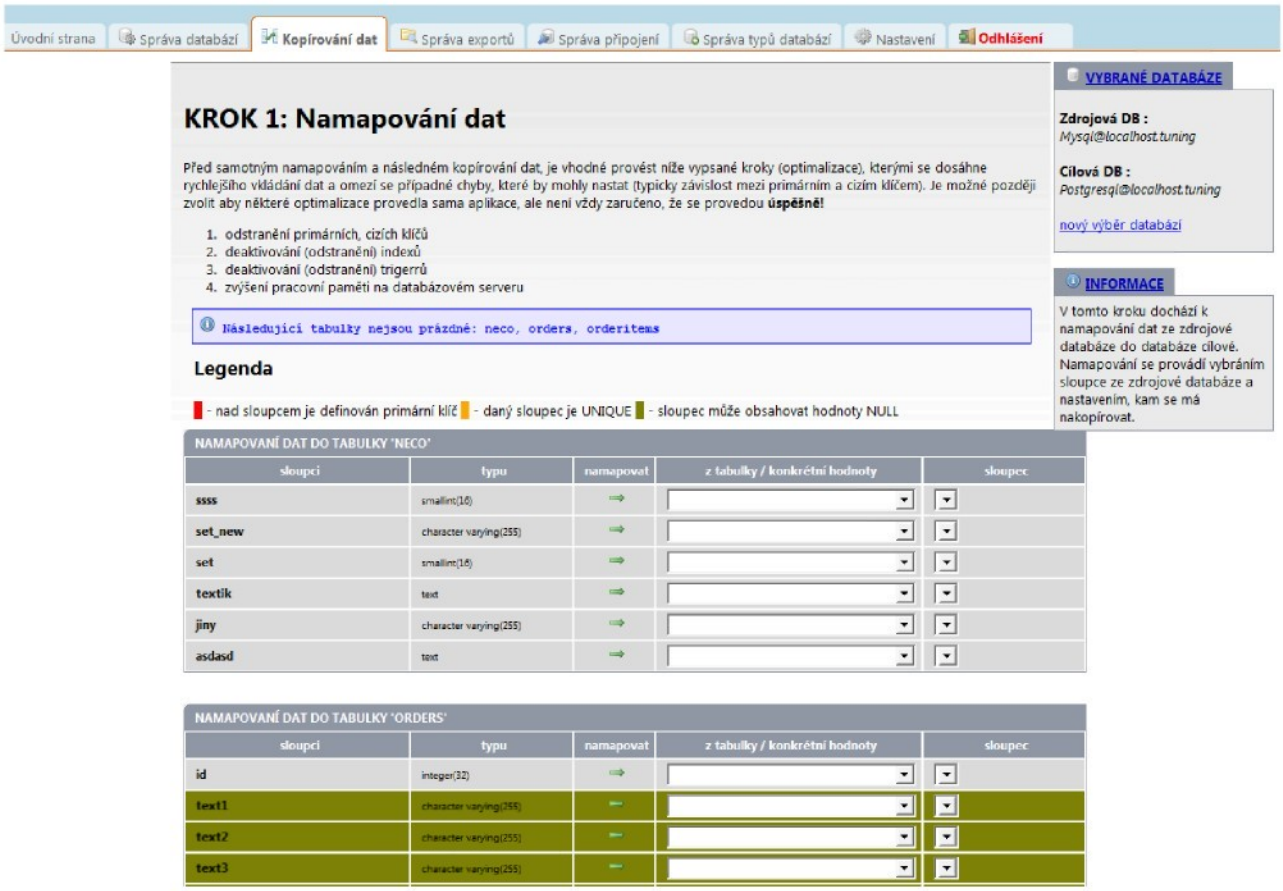
Po úspěšném přihlášení do aplikace se zobrazí úvodní stránka. Následným vybráním položky *„Kopírování dat"* v menu se zobrazí stránka s formulářem (obrázek č. 13) pro výběr zdrojové a cílové databáze. Po kliknutí na tlačítko *„Nastavit výběr"* se vše uloží a opětovným vybráním položky *„Kopírování dat"* lze přistoupit k prvnímu kroku.



obrázek č. 13 – výběr zdrojové a cílové databáze

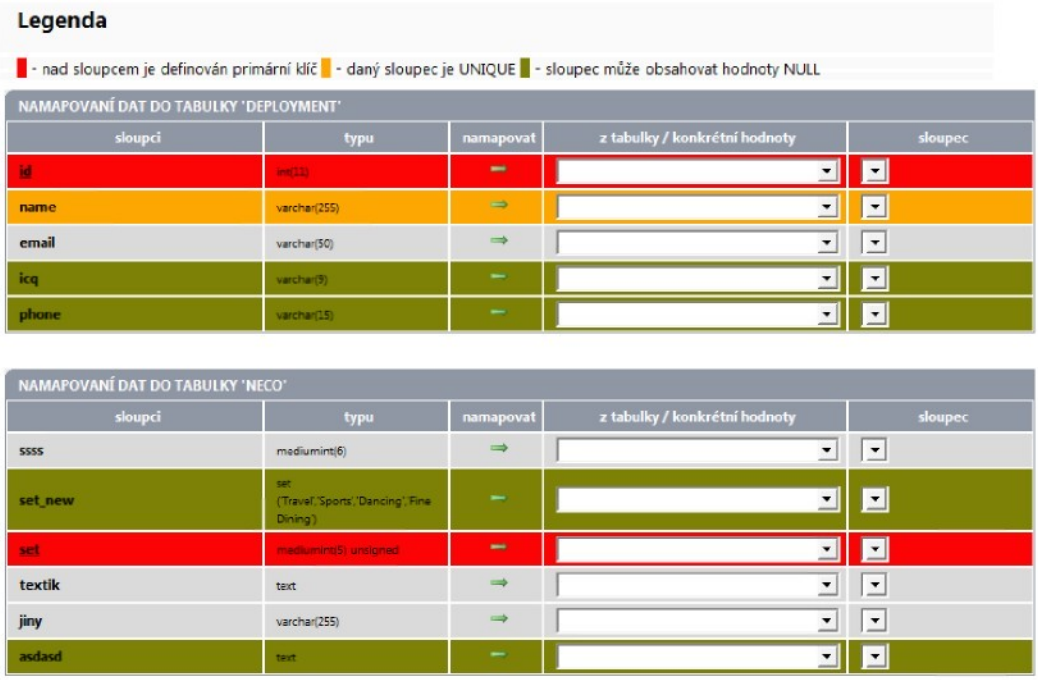
6.1. KROK 1 – NAMAPOVÁNÍ DAT

Před samotným mapováním dat je kontrolováno, zda jsou všechny tabulky v cílové databázi prázdné a v případě, že některé nejsou, je o tom uživatel patřičně informován (obrázek č. 14). Tato kontrola je v aplikaci z toho důvodu, že chceme ve většině případů kopírovat data do prázdných tabulek.



obrázek č. 14 – první krok

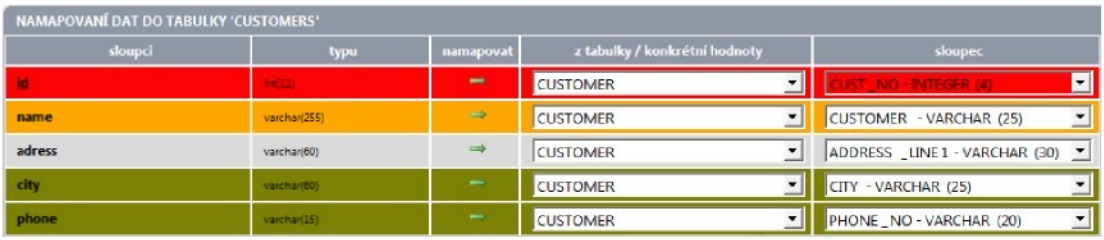
Jak bylo napsáno v kapitole Přenos dat mezi databázemi, kopírování dat by mělo probíhat do čistých tabulek. Na to se však nelze stoprocentně spolehnout, proto se na stránce nacházejí informace o optimalizacích, které by měly být na začátku provedeny. Je to i důvod proč aplikace ověřuje, zda jsou nad sloupci definovány primární klíče, unikátní indexy a zda mohou sloupce obsahovat hodnoty `NULL`. Tyto sloupce jsou pak barevně odlišeny od ostatních, což je nejjednodušší a nejpřehlednější způsob jak informovat uživatele. Ten tak na první pohled vidí, o jaké sloupce jde. Barvy byly voleny tak, aby byly jasně rozlišitelné. Součástí je i legenda, která informuje o tom, co která barva znamená. Ukázka je na obrázku č. 15.



obrázek č. 15 – barevné rozlišení sloupců

Na výše zobrazeném obrázku je možné vidět dvě tabulky z cílové databáze s vypsanými sloupci a informacemi o jejich typech (s maximální přípustnou délkou). Tyto informace se nachází v XHTML tabulkách, konkrétně v jejich levé části. Pravá část slouží k namapování dat.

Mapování probíhá výběrem tabulky ze zdrojové databáze a následným výběrem sloupce, kde sloupce jsou opět barevně rozlišovány. V některých případech pak uživatel vidí, že vybral sloupec se stejnými vlastnostmi (například unikátnost všech záznamů). Tyto vlastnosti jsou ovšem dostupné pouze v prohlížečích Opera a Internet Explorer. Ostatní vyhovující prohlížeče zobrazí barvu pouze při kliknutí na formulářový prvek `SELECT` a po následném vybrání sloupce nastaví barvu na bílou. Výsledek namapování jedné tabulky by pak mohl vypadat jako tomu je na obrázku č. 16.



obrázek č. 16 – namapovaná data do tabulky

Kromě možnosti výběru ze zdrojových tabulek a následně příslušných sloupců, lze také vybrat konkrétní hodnoty, které budou do sloupce vloženy. Příkladem může být přiřazení hodnoty `FALSE` sloupci určujícímu, zda je uživatel přihlášen. Vybírat lze z následujících možností:

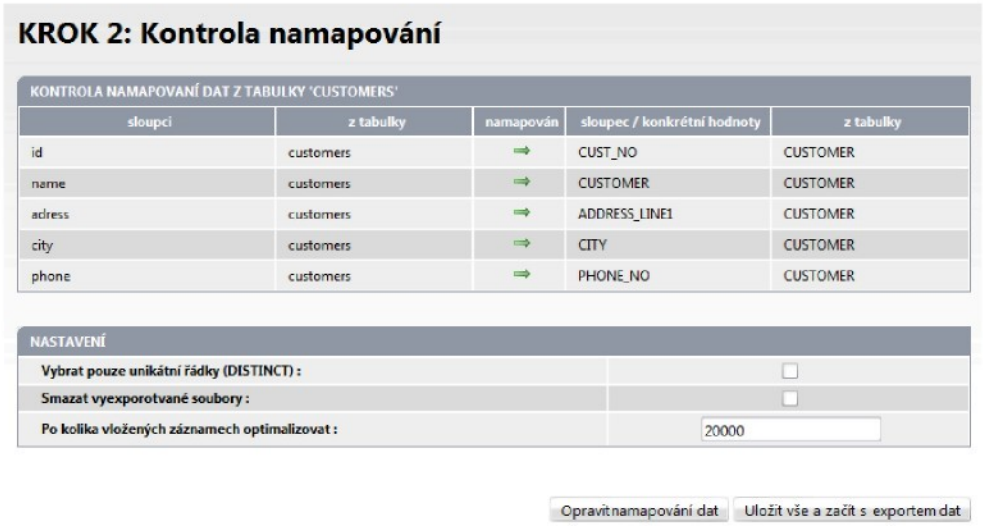
- **vložit DEFAULTní hodnoty** – vloží se do daného sloupce základní hodnoty, nadefinované při tvorbě struktury tabulky
- **vložit hodnoty NULL** – vloží se do daného sloupce hodnoty `NULL`
- **vložit hodnoty TRUE** – vloží se do daného sloupce hodnoty `TRUE`
- **vložit hodnoty FALSE** – vloží se do daného sloupce hodnoty `FALSE`
- **vložit aktuální časové/datové hodnoty** – vloží se do daného sloupce aktuální hodnota `TIMESTAMP`[7] formátovaná dle typu daného sloupce (např. hodnota `TIMESTAMP` se přeformátuje na typ sloupce `DATETIME`, tedy do podoby rok-měsíc-den hodina:minuta:sekunda)
- **vložit řetězec** – definuje se řetězec, který se vloží do daného sloupce
- **vložit číslo** – definuje se číslo, které se vloží do daného sloupce
- **Vložit unikátní ID (číslo)** – definuje se počáteční číslo, od kterého se bude začínat, poté dochází k postupné inkrementaci, vždy o jedničku

[7] počet sekund, které uběhly od 1 Ledna, 1970

6.2. KROK 2 – KONTROLA NAMAPOVÁNÍ

Po kliknutí na tlačítko *„Pokračovat“* se aplikace přesune k druhému kroku. V něm dojde k rekapitulaci a následné kontrole namapování uživatelem.

Zobrazuje se také formulář s nastavením. V něm se nacházejí tři volby, z nichž poslední se týká optimalizací. Touto hodnotou se především určí, po kolika záznamech budou probíhat transakce a příkazy[8] pro optimalizaci.



obrázek č. 17 – kontrola namapování

6.3. KROK 3 – EXPORT DAT

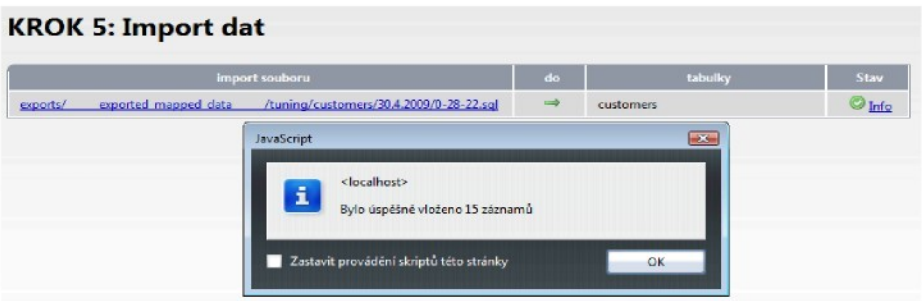
Po kliknutí na tlačítko *„Uložit vše a začít s exportem dat“* se aplikace přesune ke třetímu kroku. Zobrazí se tabulka se seznamem cílových tabulek, pro které jsou exportována data s počtem záznamů a aktuálním stavem. V tomto kroku je pomocí AJAXu volán skript pro export dat, vždy pro jednu tabulku. Po jeho skončení je zavolán skript znovu, ale tentokrát pro jinou tabulku. Výsledek je na následujícím obrázku č. 18.

[8] například VACUUM pro PostgreSQL

obrázek č. 18 – úspěšně exportovaná data pro tabulku

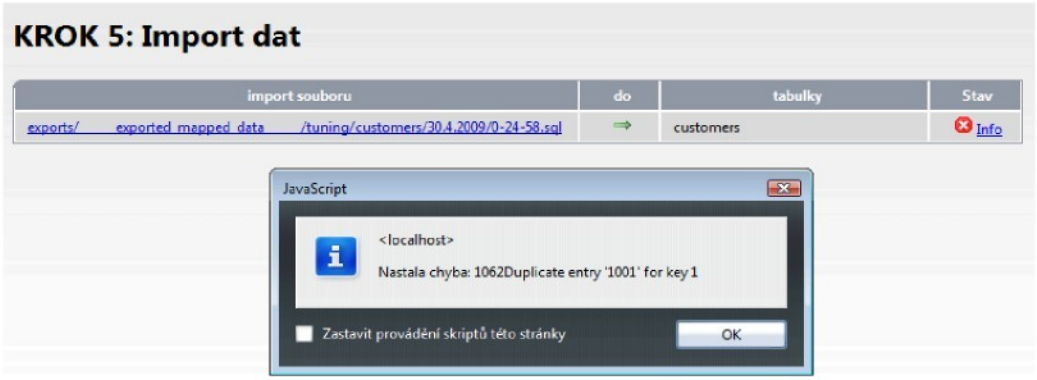
6.4. KROK 4 – IMPORT DAT

Po kliknutí na tlačítko *„Pokračovat importováním vyexportovaných dat"* se aplikace přesune k poslednímu kroku. Zde je opět využito AJAXu a dochází k volání skriptu pro import dat, pouze pro jednu tabulku jako v případě exportu. Pokud je import proveden úspěšně, může si uživatel po kliknutí na odkaz v tabulce stáhnout vyexportovaný soubor a po kliknutí na tlačítko *„Info"* zjistí, kolik bylo úspěšně vloženo záznamů (obrázek č. 19).



obrázek č. 19 – úspěšně importována data do tabulky

V případě neúspěšného importu dat, se po kliknutí na tlačítko *„Info"* zobrazí informace o tom, k jaké chybě došlo (obrázek č. 20).



obrázek č. 20 – neúspěšné importování dat do tabulky

7. ZÁVĚR

Výsledná aplikace se skládá z 37 souborů se skripty a 49 šablon. Všechny soubory obsahují téměř 10 000 řádků kódu. Program splňuje nejen všechny definované cíle, ale také nabízí další vlastnosti a funkce. Především jde o import dat ze souborů ve formátech XML, CSV a SQL s možností namapování dat do příslušných sloupců v tabulce, export dat do souborů ve formátech XML, CSV, MHT a SQL s možností výběru konkrétních sloupců, které budou exportovány, možnost exportovat, importovat a kopírovat tabulky s velkým počtem záznamů a spravovat připojení k databázím.

Aplikace nabízí komfortní práci především díky vlastní abstraktní databázové vrstvě, AJAXu, přehlednosti, nastavitelnosti a možnosti pracovat i s velkým množstvím dat při importu, exportu a samotném kopírování.

Následující vývoj aplikace bude záviset především na potřebách společnosti CLWEB. Na základě jejich podnětů se aplikace bude dále rozvíjet a doplňovat o další funkcionality.

Zlepšování aplikace by se dále mohlo nést v duchu přenosu dat. Zde existuje dostatečný prostor pro její další rozvoj. Pro lepší práci, by bylo dobré vhodně navrhnout a implementovat modul, který by umožnil převod mezi různými datovými typy. Za pozornost by také stálo rozšíření podporovaných databází o Microsoft SQL Server a umožnit ve správě databází tvorbu a úpravu databázové struktury tabulek.

8. ZDROJE INFORMACÍ

[1] Gutmans Andi, Bakken Stig Saether, Rethans Derick, Mistrovství v PHP 5, ISBN 80-251-0799-X, Computer Press, 2005

[2] Luboslav Lacko, Web a databáze - programujeme internetové aplikace, ISBN 80-7226-555-5, Computer Press, 2001

[3] dokumentace k PHP

http://www.php.net/docs.php

[4] Stránky o PHP programování

http://php.vrana.cz/

[5] Stránky o PHP programování a databázích

http://www.alberton.info/

[6] Oficiální stránky PostgreSQL databáze

http://www.postgresql.org/

[7] Oficiální stránky MySQL databáze

http://www.mysql.org/

[8] Oficiální stránky Firebird databáze

http://www.firebirdsql.org/

PŘÍLOHY

A. Obsah přiloženého CD

Přiložený CD ROM v jednotlivých složkách obsahuje:

- **AplikaceX2Y** – zdrojové kódy aplikace X2Y
- **DiplomovaPrace** – adresář s touto diplomovou prací ve formátu pdf a doc
- **DokumentaceAPI** – dokumentace k API vytvořeného systému
- **UzivatelskyManual** – adresář obsahující uživatelský manuál, popisující jednotlivé části aplikace X2Y


X2Y system

SYSTEM

SYSTEM

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N2612 - Elektrotechnika a informatika
Studijní obor: 1802T007 - Informační technologie

UNIVERZÁLNÍ WEBOVÁ APLIKACE NA PŘEVOD DAT MEZI DATABÁZEMI

UNIVERSAL WEB APLICATION FOR TRANSFERING DATA BETWEEN DATABASES

Uživatelský manuál

Autor: Bc. Jan Kahoun

V Liberci 25. 5. 2009

OBSAH

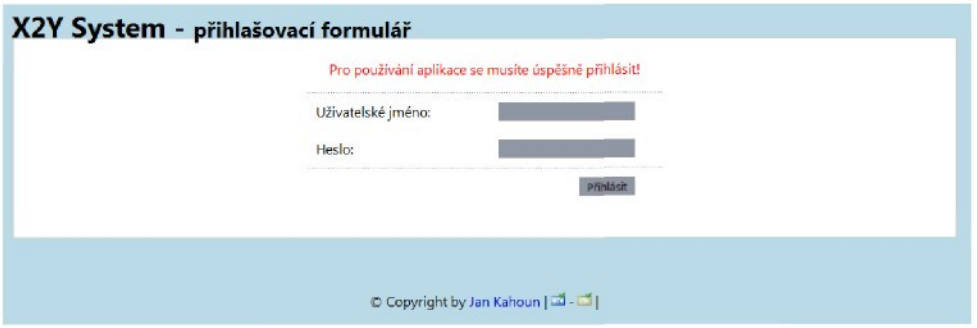
1. ÚVOD

Tato uživatelská příručka popisuje pouze jak správně pracovat s webovou aplikací X2Y. Je součástí diplomové práce *„UNIVERZÁLNÍ WEBOVÁ APLIKACE NA PŘEVOD DAT MEZI DATABÁZEMI“*, kde se nachází podrobný popis, jak je aplikace navržena. Dokumentace k API aplikace je na přiloženém CD.

2. PŘIHLAŠOVÁNÍ A ODHLAŠOVÁNÍ UŽIVATELE

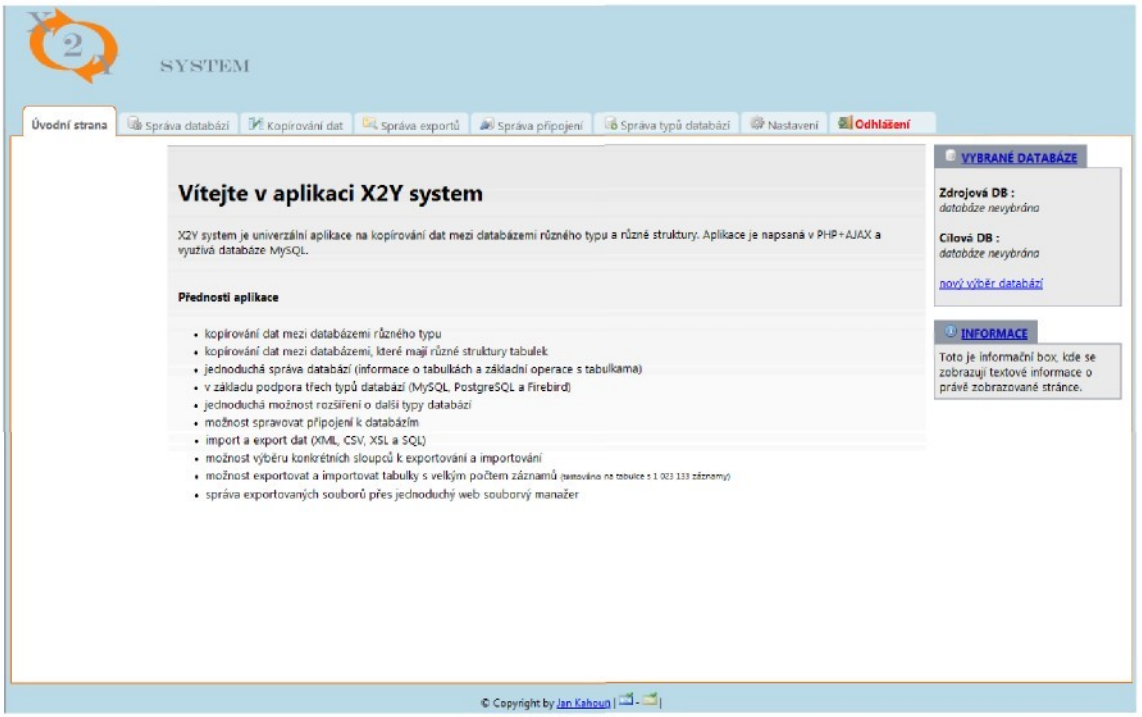
2.1. Přihlášení uživatele

Před samotným používáním aplikace je nutné se úspěšně přihlásit (obrázek č. 1). Pokud se jedná o první přihlášení nebo je-li použito základní nastavení, je přihlašovací jméno nastaveno na *admin* a heslo na *x2y09*.



obrázek č. 1 – přihlašovací formulář

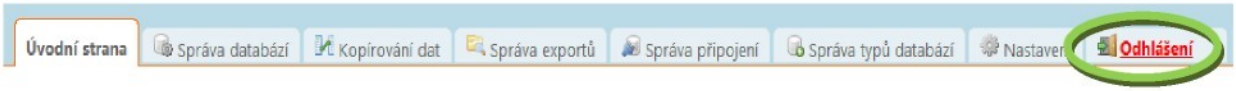
Po úspěšném přihlášení do aplikace se zobrazí úvodní stránka, kde je stručný popis aplikace a výpis její vlastností.



obrázek č. 2 - úvodní obrazovka po přihlášení

2.2. Odhlášení uživatele

Odhlášení uživatele z aplikace proběhne po kliknutí na poslední odkaz „*Odhlášení*“ v hlavním menu. Odkaz pro dohlášení je odlišen od ostatních tučně červeným písmem, jak si lze všimnout na obrázku č. 3.



obrázek č. 3 – odkaz na odhlášení v hlavním menu

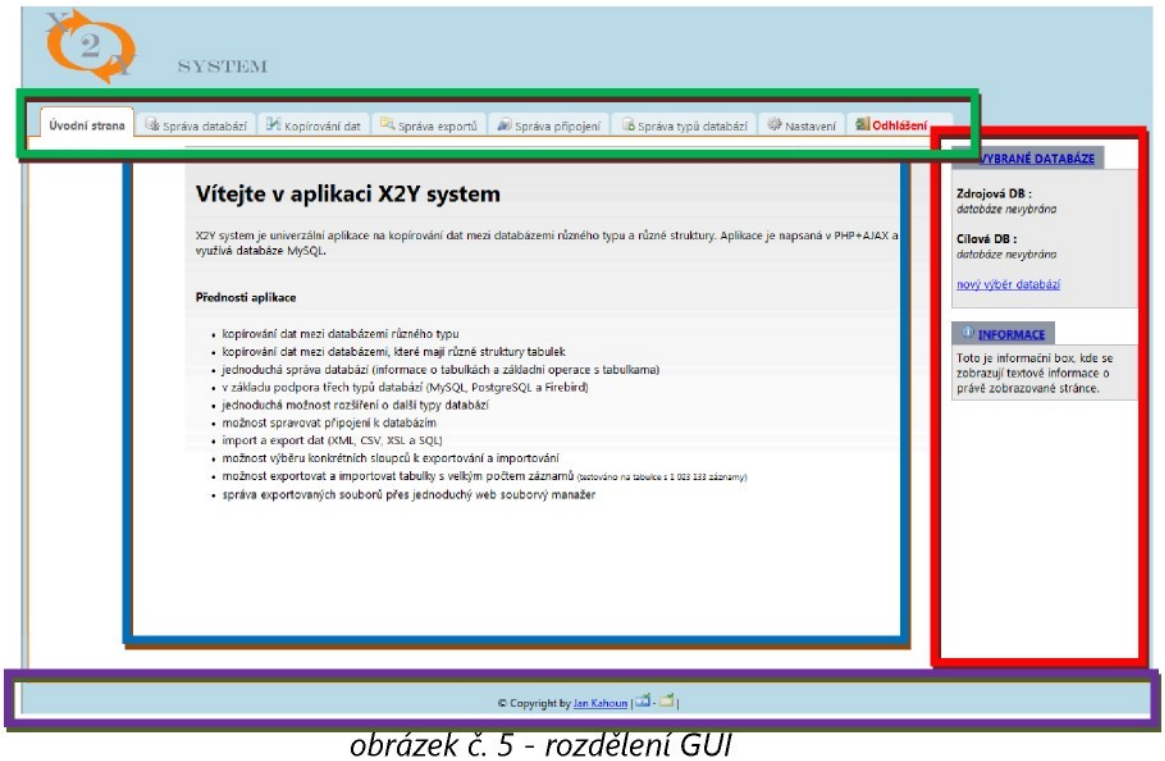
Při úspěšném odhlášení je uživatel informován, že vše proběhlo úspěšně a je mu nabídnuta možnost znovu se přihlásit (obrázek č. 3).

obrázek č. 4 - úspěšné odhlášení uživatele

3. GUI (GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ)

Uživatelské rozhraní je v aplikaci rozděleno na čtyři hlavní části:

- **pracovní** – uživateli se zde zobrazují formuláře, data a informace nutné pro jeho práci s aplikací (na obrázku č. 5 modrý rámeček)
- **informační** – uživateli se zde zobrazují informace o stránce, na které se aktuálně nachází a o aktuálně vybraných databázích (na obrázku č. 5 červený rámeček)
- **ladicí** – spodní část GUI, kde se v případě zapnutého ladicího módu, zobrazují dodatečné informace uživateli (na obrázku č. 5 zelený rámeček)
- **hlavní menu** – seznam dostupných odkazů na hlavní části aplikace (na obrázku č. 5 fialový rámeček)



obrázek č. 5 - rozdělení GUI

4. HLAVNÍ ČÁSTI APLIKACE

Seznam hlavních částí aplikace, lze najít v hlavním menu (obrázek č. 6), které obsahuje jejich seznam v podobě odkazů. Skládá se z následujících částí:

- **Úvodní strana** - stručný popis aplikace a výpis její vlastností
- **Správa databází** - zobrazování informací o tabulkách v databázích (seznam sloupců a jejich vlastnosti, definované klíče a indexy), možností provedení základních operací na jednotlivých tabulkách (odstranění a vyprázdnění), import a export dat v různých formátech a možnost provedení SQL dotazu na zdrojovou nebo cílovou databázi (viz kapitola 4.1 - Správa databází)
- **Kopírování dat** – přenos dat mezi databázemi (viz Kapitola *4.1* -

Správa databází vyžaduje, aby byly nastavené databáze, se kterými má aplikace pracovat (více informací v kapitole 5 - Výběr zdrojové a cílové databáze).

Po kliknutí na odkaz *„Správa databází"* se zobrazí v pracovní části na levé straně seznam dostupných tabulek pro cílovou a zdrojovou databázi. Na pravé straně se pak nachází dvě tabulky, jedna pro cílovou a druhá pro zdrojovou databázi. Každá z nich obsahuje seznam dostupných tabulek včetně počtu záznamů, typu (použito u MySQL k rozlišení tabulek MyISAM a InnoDB) a porovnávání, které je na dané tabulce použito (obrázek č. 7).

obrázek č. 7 – seznam dostupných tabulek v databázích

Při kliknutí na název tabulky (ať už v levé nebo pravé části) se zobrazí stránka, kde se nachází následující informace o tabulce (obrázek č. 8):

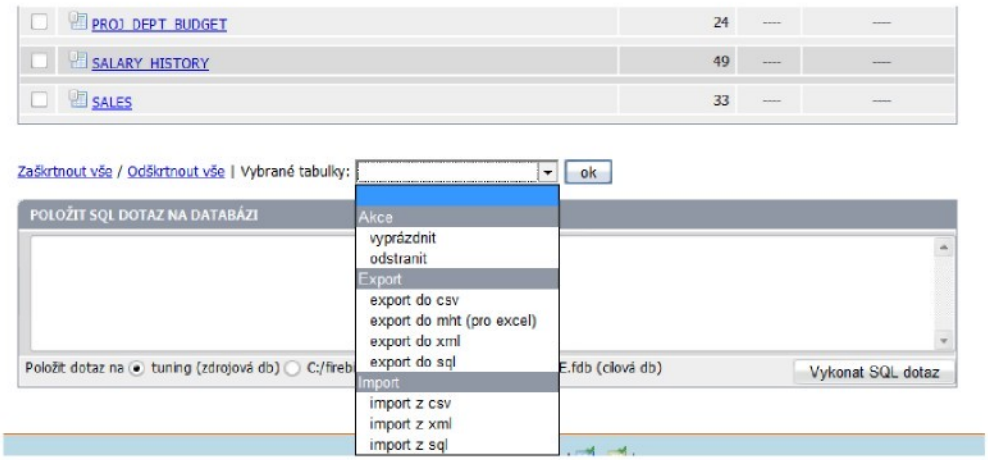
- **Sloupce tabulky**
  - Název – název sloupce
  - Typ – typ sloupce (např. mediumint(6), varchar(255), ...)
  - Porovnávání – porovnávání, které je použito pro daný sloupec (např. latin2_czech_cs, ...)
  - Nulový – zda sloupec může obsahovat hodnoty *NULL*
  - Unikátní – zda se ve sloupci mohou objevit duplikátní hodnoty
  - Výchozí – zda je pro sloupec nastavena nějaká výchozí hodnota (aplikuje se při vkládání dat)
- **Indexy**
  - Název – název indexu (pod jakým názvem je v databázi definován)
  - Typ – typ indexu (např. primary, unique, ...)
  - Mohutnost
  - Sloupec – nad jakým sloupcem či sloupci je index definován
- **Statistika řádků**
  - Porovnávání - porovnávání, které je použito pro danou tabulku
  - Řádků – celkový počet záznamů v tabulce
  - Další Autoindex (pouze MySQL) – hodnota dalšího unikátního id v tabulce (může být pouze na jednom sloupci)
  - Vytvoření (pouze MySQL) – datum vytvoření tabulky
  - Poslední změna (pouze MySQL) – datum posledná změny struktury tabulky

obrázek č. 8 - výpis informací o tabulce

Pod seznamem dostupných tabulek v databázích se nachází možnost vybrat všechny tabulky (odkaz *„Zaškrtnout vše"*), či zrušit aktuální výběr (odkaz *„Odškrtnout vše"*). Vedle je možné vybrat jednu z následujících operací, která bude provedena (obrázek č. 9):

- **Akce**
  - vyprázdnit
  - odstranit

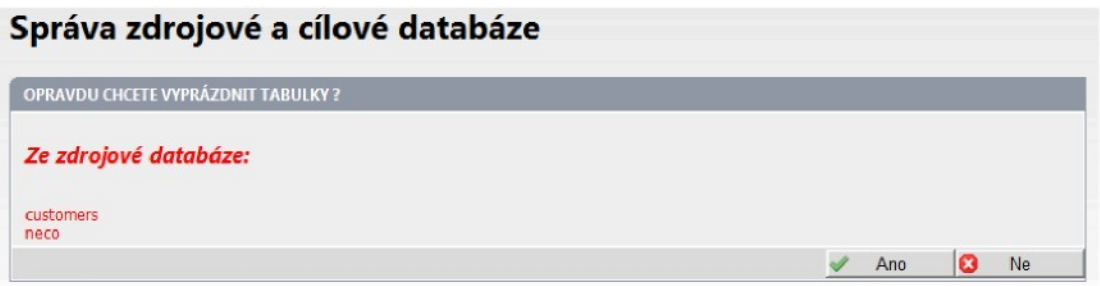
- **Export**
  - export do csv
  - export do mht (pro Excel)
  - export do xml
  - export do sql
- **Import**
  - import z csv
  - import z xml
  - import z sql



obrázek č. 9 – seznam dostupných operací

4.1.1. Akce

Jelikož *Akce* jsou nebezpečné operace, je před jejím vykonáním uživatel patřičně informován (obrázek č. 10). Teprve po kliknutí na tlačítko *„Ano"* dojde k provedení operace, v opačném případě k návratu na předchozí stránku.



obrázek č. 10 - dotaz na potvrzení provedení operace

4.1.2. Export

Při výběru exportu dat do jakéhokoliv formátu dojde k zobrazení hlavního nastavení (obrázek č. 11). V něm je možné nastavit následující vlastnosti:

- **Výběr jednotlivých sloupců** – před exportem, bude umožněno vybrat, jaké sloupce budou exportovány
- **Vybrat pouze unikátní řádky (DISTINCT)** – při exportu dat dojde k vybrání pouze unikátních záznamů
- **Kódování dat** – výběr v jakém kódování budou data uložena (UTF-8, Windows-1250 a ISO 8859-2)

obrázek č. 11 - hlavní nastavení při exportu

Formát CSV

Tento formát umožňuje dále nastavit:

- **Oddělovač –** čím budou data oddělena (např. středník, tabelátor)
- **Data uzavřená v –** v čem budou data uzavřena (např. jednoduché nebo dvojité uvozovky, ...)
- **Escapovací znak –** znak, který bude použit v datech před znaky, v nichž jsou data uzavřena (např. data jsou uzavřená v „“ a escapovací znak je \ -> výsledkem bude něco podobného: `„toto je citace: \"Ahoj světe!\" nějaký text“`)
- **Jména sloupců na prvním řádku** - na prvním řádku budou názvy sloupců na dalších už budou jen data

Formát MHT

Tento formát umožňuje dále nastavit:

- **Jména sloupců na prvním řádku** – na prvním řádku budou názvy sloupců na dalších už budou jen data
- **Zapnout podporu pro Excel 2003** – zapíná podporu exportu dat pro Excel 2003 (primárně podpora pro Excel 2007, který umožňuje mít v jednom sešitu 1 048 576 řádků oproti 65 536 řádkům v Excelu 2003)

Formát XML

Tento formát nemá další možnosti nastavení.

Formát SQL

Tento formát umožňuje dále nastavit:

- **Úplné inserty** – názvy sloupců v dotazu (např. `INSERT INTO` `` `x2y_dbs_types` `` `(` `` `dt_Id` ``, `` `dt_Name` ``, `` `dt_Installed` `` `) VALUES (1, 'mysql', 1);`)
- **Rozšířené inserty** – jeden dotaz pro všechny záznamy (např. rozšířeného insertu: `INSERT INTO` `` `x2y_dbs_types` `` `VALUES (1, 'mysql', 1), (2, 'firebird', 1), (3, 'postgresql', 1);`)

Nejsou-li vybrány *„Úplné inserty"* ani *„Rozšířené inserty"*, výsledné dotazy budou ve tvaru `INSERT INTO` `` `x2y_dbs_connections` `` `VALUES (17, 1, 'parfémy', 'localhost', 1);`.

4.1.3. Import

Při výběru importu dat do jakéhokoliv formátu dojde k zobrazení hlavního nastavení (obrázek č. 12). V něm je možné nastavit položku *„Po kolika záznamech importovat"*, která určuje, kolik záznamů bude vloženo v rámci jedné transakce.

obrázek č. 12 - hlavní nastavení při importu

Formát CSV

Tento formát umožňuje dále nastavit:

- **Oddělovač –** čím jsou data oddělena (např. středník, tabelátor)
- **Data uzavřená v –** v čem jsou data uzavřena (např. jednoduché nebo dvojité uvozovky, ...)
- **Escapovací znak –** znak, který je použit v datech před znaky, v nichž jsou data uzavřena (např. data jsou uzavřená v „" a escapovací znak je \ -> výsledkem bude něco podobného: *„toto je citace: \"Ahoj světe!\" nějaký text"*)

Formát XML

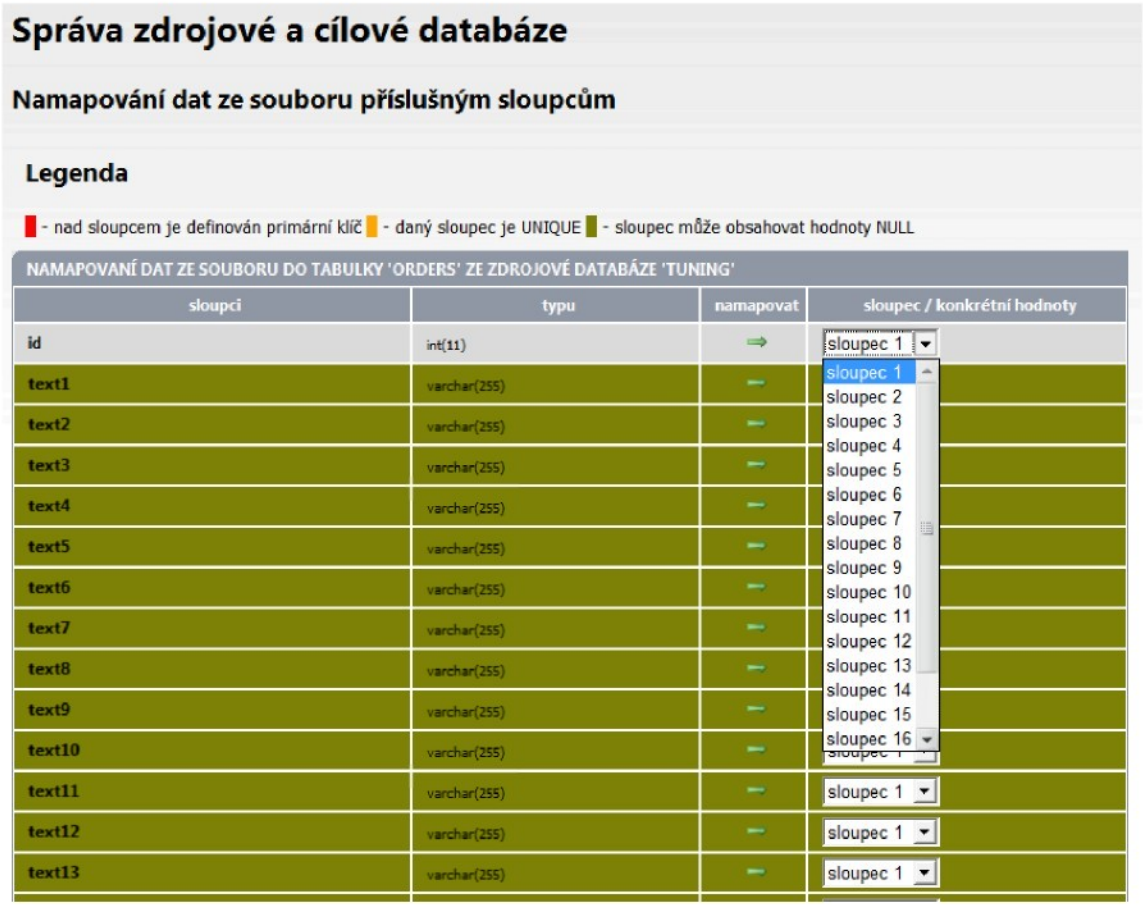
Tento formát nemá další možnosti nastavení.

Formát SQL

Tento formát nemá další možnosti nastavení.

Namapování dat při importu

Namapování dat probíhá na samostatné stránce (obrázek č. 13). Příslušným sloupcům v tabulkách jsou vybírána data (lze si je představit také jako sloupce) ze zvoleného souboru. Jednotlivé sloupce jsou aplikací barevně odlišeny (více v kapitole **Chyba! Nenalezen zdroj odkazů.** - **Chyba! Nenalezen zdroj odkazů.**)



obrázek č. 13 – mapování dat při importu

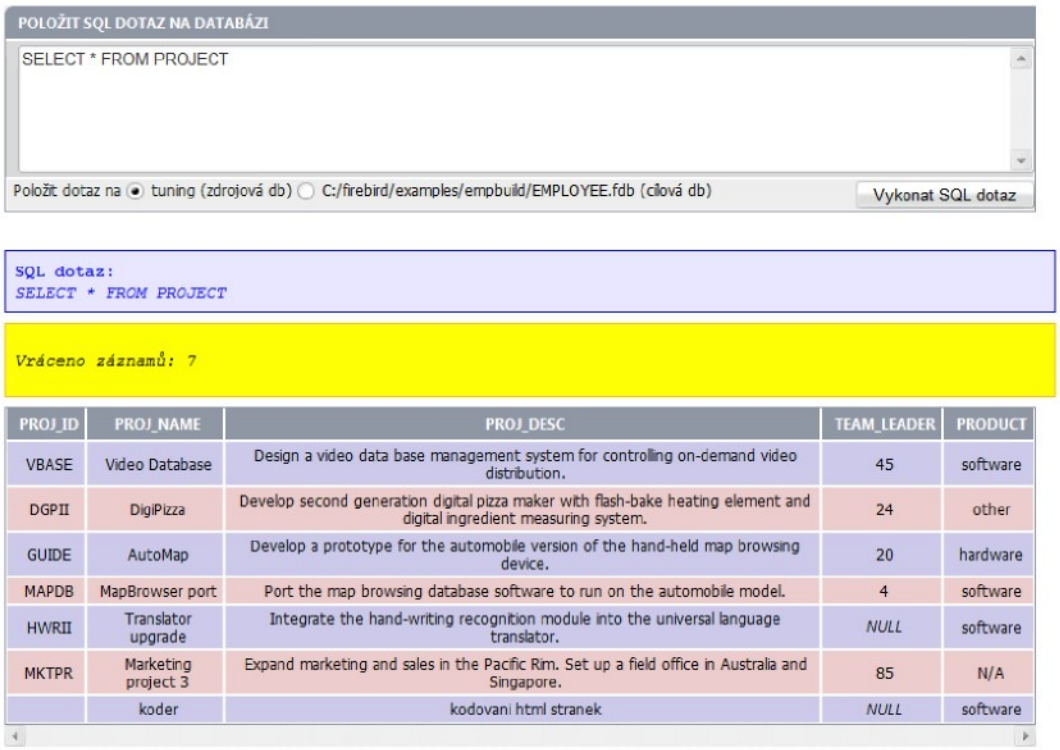
Po kliknutí na tlačítko *„Začít import dat“* dojde k importování dat, dle nastaveného namapování.

4.1.4. Formulář pro vykonání SQL dotazu

Součástí správy databází je i formulář (obrázek č. 14), do kterého je možné napsat jakýkoliv SQL dotaz a následně pak vykonat jak na cílové, tak i na zdrojové databázi.

obrázek č. 14 - formulář pro vykonání SQL dotazu

Jelikož není dotaz aplikací jakkoliv zkoumán a upravován, je nutné brát v potaz charakter dotazu, jenž bude vykonán (např. výběr všech záznamů z tabulky, kde jich je několik tisíc, by měl být omezen jen na několik desítek). Výsledek úspěšně provedeného dotazu je na obrázku č. 15.



PROJ_ID	PROJ_NAME	PROJ_DESC	TEAM_LEADER	PRODUCT
VBASE	Video Database	Design a video data base management system for controlling on-demand video distribution.	45	software
DGPII	DigiPizza	Develop second generation digital pizza maker with flash-bake heating element and digital ingredient measuring system.	24	other
GUIDE	AutoMap	Develop a prototype for the automobile version of the hand-held map browsing device.	20	hardware
MAPDB	MapBrowser port	Port the map browsing database software to run on the automobile model.	4	software
HWRII	Translator upgrade	Integrate the hand-writing recognition module into the universal language translator.	*NULL*	software
MKTPR	Marketing project 3	Expand marketing and sales in the Pacific Rim. Set up a field office in Australia and Singapore.	85	N/A
	koder	kodovani html stranek	*NULL*	software

obrázek č. 15 - výsledek provedeného dotazu

- Kopírování Dat)

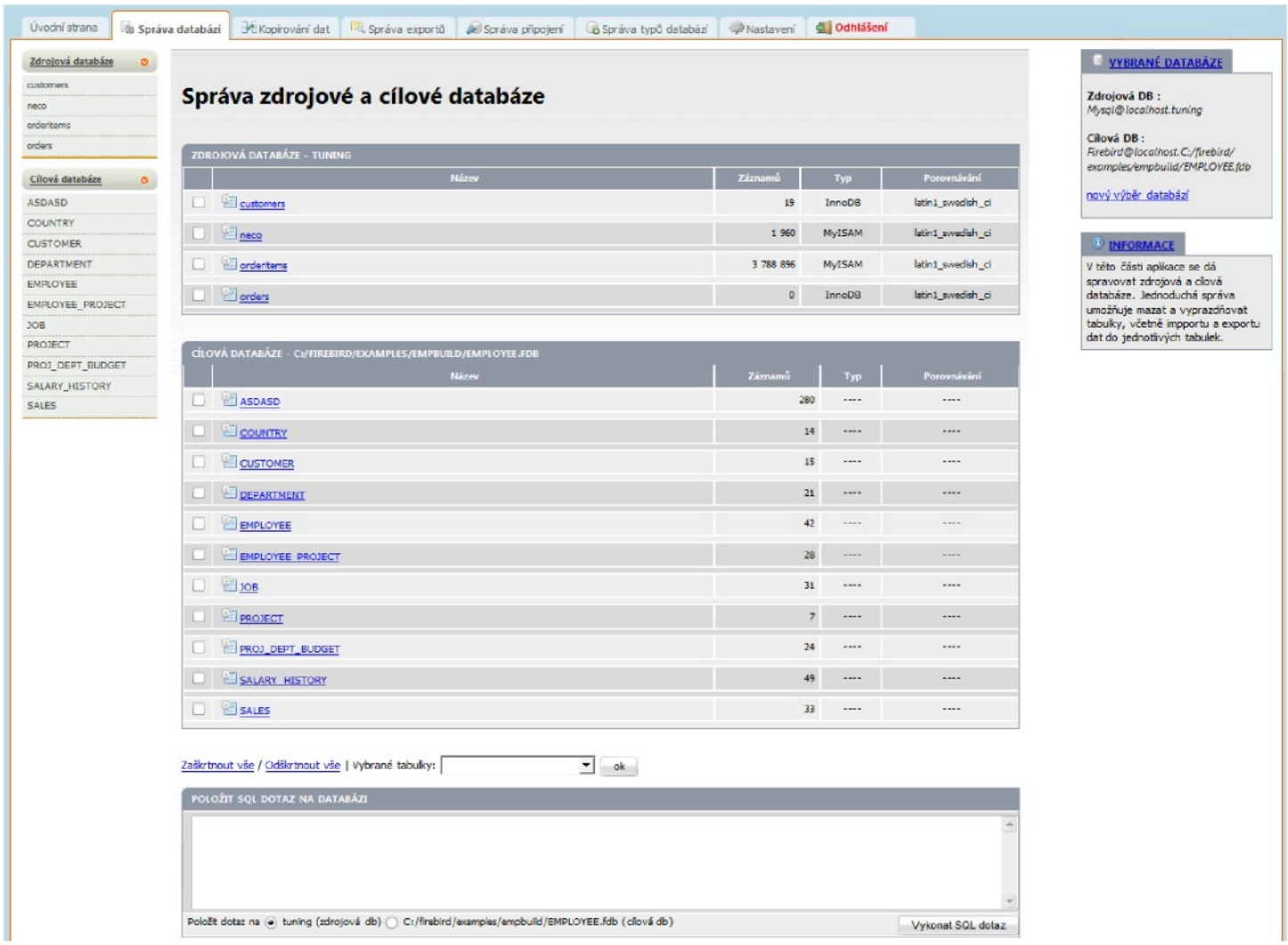
- **Správa exportů** - správa exportovaných souborů přes jednoduchý web souborový manažer (viz kapitola 4.4 - Správa exportů)
- **Správa připojení** – definování a spravování připojení k databázím (viz kapitola 4.5 - Správa připojení)
- **Správa typů databází** – správa ovladačů (driverů) pro různé typy databází (viz kapitola 4.6 - Správa typů databází (správa ovladačů))
- **Nastavení** – hlavní nastavení aplikace (viz kapitola 4.7 - Nastavení)
- **Odhlášení** – odhlášení uživatele z aplikace (viz kapitola *2.2 - Odhlášení uživatele*)

obrázek č. 6 - hlavní menu

4.2. Správa databází

Správa databází vyžaduje, aby byly nastavené databáze, se kterými má aplikace pracovat (více informací v kapitole 5 - Výběr zdrojové a cílové databáze).

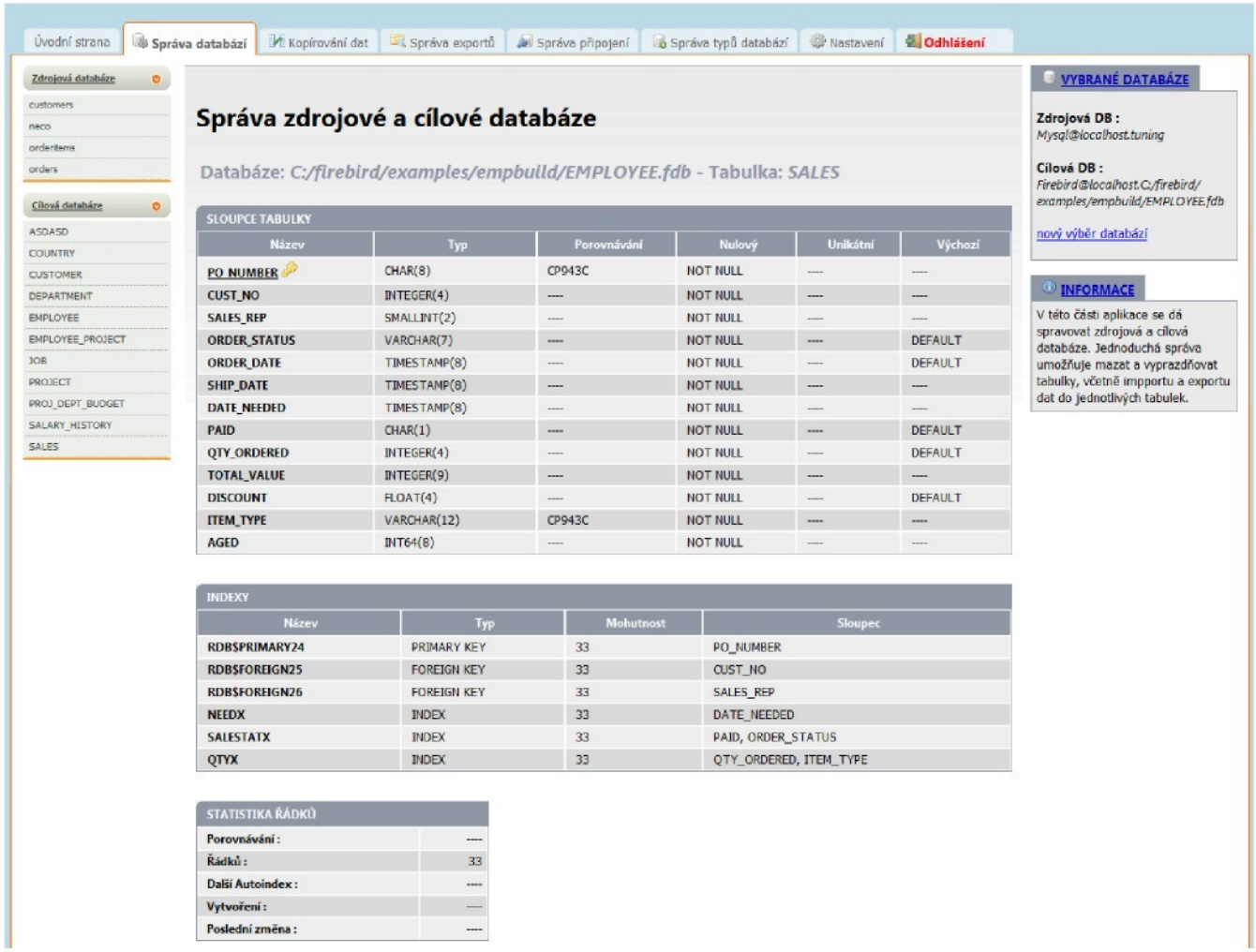
Po kliknutí na odkaz *„Správa databází“* se zobrazí v pracovní části na levé straně seznam dostupných tabulek pro cílovou a zdrojovou databázi. Na pravé straně se pak nachází dvě tabulky, jedna pro cílovou a druhá pro zdrojovou databázi. Každá z nich obsahuje seznam dostupných tabulek včetně počtu záznamů, typu (použito u MySQL k rozlišení tabulek MyISAM a InnoDB) a porovnávání, které je na dané tabulce použito (obrázek č. 7).



obrázek č. 7 – seznam dostupných tabulek v databázích

Při kliknutí na název tabulky (ať už v levé nebo pravé části) se zobrazí stránka, kde se nachází následující informace o tabulce (obrázek č. 8):

- **Sloupce tabulky**
  - Název – název sloupce
  - Typ – typ sloupce (např. mediumint(6), varchar(255), ...)
  - Porovnávání – porovnávání, které je použito pro daný sloupec (např. latin2_czech_cs, ...)
  - Nulový – zda sloupec může obsahovat hodnoty *NULL*
  - Unikátní – zda se ve sloupci mohou objevit duplikátní hodnoty
  - Výchozí – zda je pro sloupec nastavena nějaká výchozí hodnota (aplikuje se při vkládání dat)
- **Indexy**
  - Název – název indexu (pod jakým názvem je v databázi definován)
  - Typ – typ indexu (např. primary, unique, ...)
  - Mohutnost
  - Sloupec – nad jakým sloupcem či sloupci je index definován
- **Statistika řádků**
  - Porovnávání - porovnávání, které je použito pro danou tabulku
  - Řádků – celkový počet záznamů v tabulce
  - Další Autoindex (pouze MySQL) – hodnota dalšího unikátního id v tabulce (může být pouze na jednom sloupci)
  - Vytvoření (pouze MySQL) – datum vytvoření tabulky
  - Poslední změna (pouze MySQL) – datum posledná změny struktury tabulky

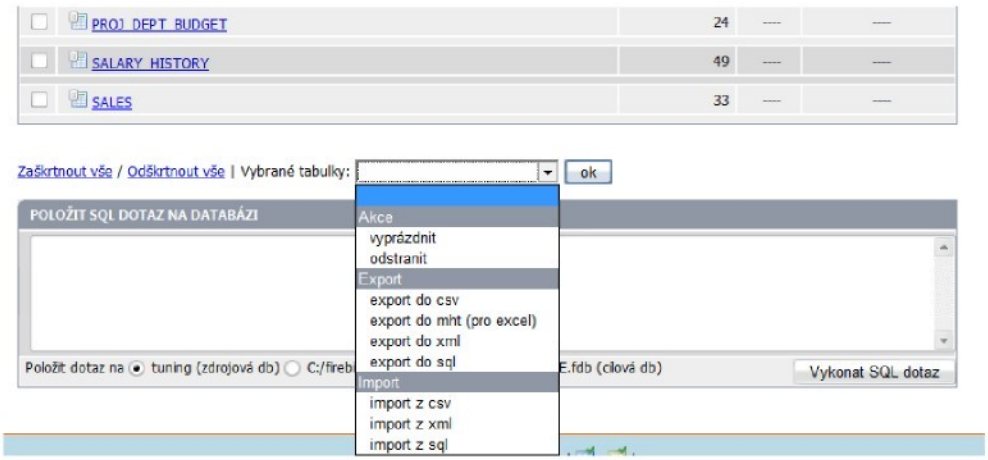


obrázek č. 8 - výpis informací o tabulce

Pod seznamem dostupných tabulek v databázích se nachází možnost vybrat všechny tabulky (odkaz *„Zaškrtnout vše“*), či zrušit aktuální výběr (odkaz *„Odškrtnout vše“*). Vedle je možné vybrat jednu z následujících operací, která bude provedena (obrázek č. 9):

- **Akce**
  - vyprázdnit
  - odstranit

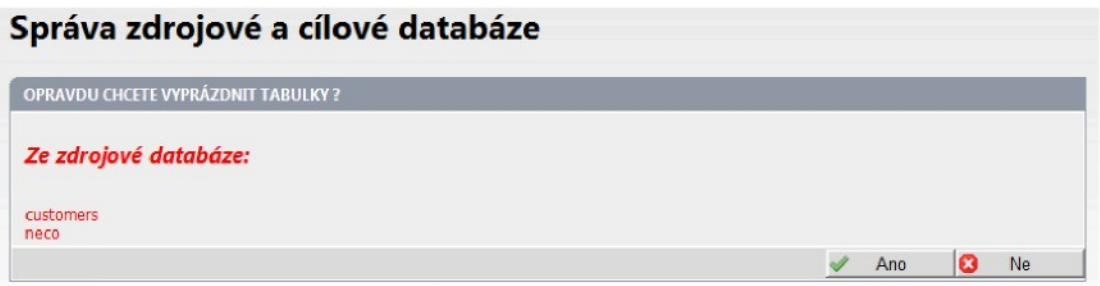
- **Export**
  - export do csv
  - export do mht (pro Excel)
  - export do xml
  - export do sql
- **Import**
  - import z csv
  - import z xml
  - import z sql



obrázek č. 9 – seznam dostupných operací

4.2.1. Akce

Jelikož *Akce* jsou nebezpečné operace, je před jejím vykonáním uživatel patřičně informován (obrázek č. 10). Teprve po kliknutí na tlačítko *„Ano"* dojde k provedení operace, v opačném případě k návratu na předchozí stránku.

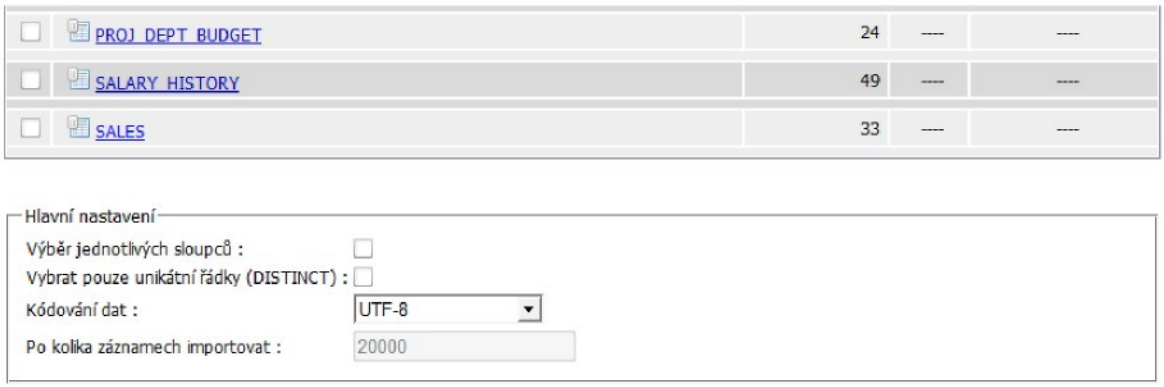


obrázek č. 10 - dotaz na potvrzení provedení operace

4.2.2. Export

Při výběru exportu dat do jakéhokoliv formátu dojde k zobrazení hlavního nastavení (obrázek č. 11). V něm je možné nastavit následující vlastnosti:

- **Výběr jednotlivých sloupců** – před exportem, bude umožněno vybrat, jaké sloupce budou exportovány
- **Vybrat pouze unikátní řádky (DISTINCT)** – při exportu dat dojde k vybrání pouze unikátních záznamů
- **Kódování dat** – výběr v jakém kódování budou data uložena (UTF-8, Windows-1250 a ISO 8859-2)



obrázek č. 11 - hlavní nastavení při exportu

Formát CSV

Tento formát umožňuje dále nastavit:

- **Oddělovač –** čím budou data oddělena (např. středník, tabelátor)
- **Data uzavřená v –** v čem budou data uzavřena (např. jednoduché nebo dvojité uvozovky, ...)
- **Escapovací znak –** znak, který bude použit v datech před znaky, v nichž jsou data uzavřena (např. data jsou uzavřená v „“ a escapovací znak je \ -> výsledkem bude něco

podobného: *„toto je citace: \"Ahoj světe!\" nějaký text"*)

- **Jména sloupců na prvním řádku** - na prvním řádku budou názvy sloupců na dalších už budou jen data

Formát MHT

Tento formát umožňuje dále nastavit:

- **Jména sloupců na prvním řádku** – na prvním řádku budou názvy sloupců na dalších už budou jen data
- **Zapnout podporu pro Excel 2003** – zapíná podporu exportu dat pro Excel 2003 (primárně podpora pro Excel 2007, který umožňuje mít v jednom sešitu 1 048 576 řádků oproti 65 536 řádkům v Excelu 2003)

Formát XML

Tento formát nemá další možnosti nastavení.

Formát SQL

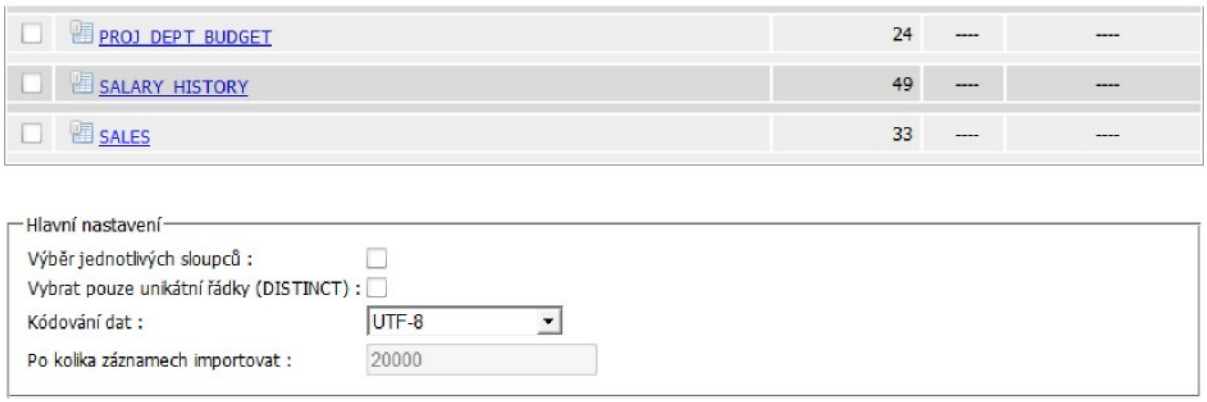
Tento formát umožňuje dále nastavit:

- **Úplné inserty** – názvy sloupců v dotazu (např. *INSERT INTO `x2y_dbs_types` (`dt_Id`, `dt_Name`, `dt_Installed`) VALUES (1, 'mysql', 1);*)
- **Rozšířené inserty** – jeden dotaz pro všechny záznamy (např. rozšířeného insertu: *INSERT INTO `x2y_dbs_types` VALUES (1, 'mysql', 1), (2, 'firebird', 1), (3, 'postgresql', 1);)*

Nejsou-li vybrány *„Úplné inserty"* ani *„Rozšířené inserty"*, výsledné dotazy budou ve tvaru *INSERT INTO `x2y_dbs_connections` VALUES (17, 1, 'parfémy', 'localhost', 1);.*

4.2.3. Import

Při výběru importu dat do jakéhokoliv formátu dojde k zobrazení hlavního nastavení (obrázek č. 12). V něm je možné nastavit položku *„Po kolika záznamech importovat“*, která určuje, kolik záznamů bude vloženo v rámci jedné transakce.



obrázek č. 12 - hlavní nastavení při importu

Formát CSV

Tento formát umožňuje dále nastavit:

- **Oddělovač –** čím jsou data oddělena (např. středník, tabelátor)
- **Data uzavřená v –** v čem jsou data uzavřena (např. jednoduché nebo dvojité uvozovky, …)
- **Escapovací znak –** znak, který je použit v datech před znaky, v nichž jsou data uzavřena (např. data jsou uzavřená v „“ a escapovací znak je \ -> výsledkem bude něco podobného: *„toto je citace: \“Ahoj světe!\“ nějaký text“*)

Formát XML

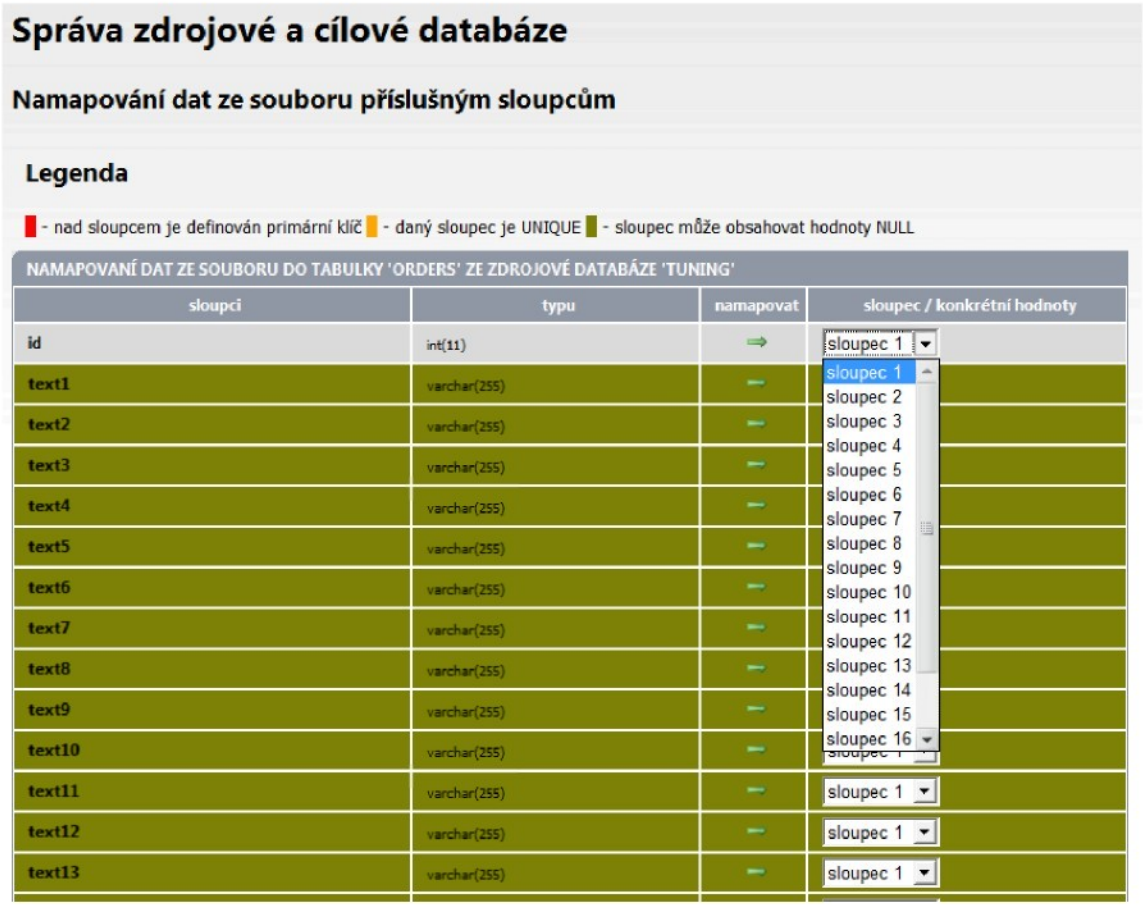
Tento formát nemá další možnosti nastavení.

Formát SQL

Tento formát nemá další možnosti nastavení.

Namapování dat při importu

Namapování dat probíhá na samostatné stránce (obrázek č. 13). Příslušným sloupcům v tabulkách jsou vybírána data (lze si je představit také jako sloupce) ze zvoleného souboru. Jednotlivé sloupce jsou aplikací barevně odlišeny (více v kapitole **Chyba! Nenalezen zdroj odkazů.** - **Chyba! Nenalezen zdroj odkazů.**)



obrázek č. 13 – mapování dat při importu

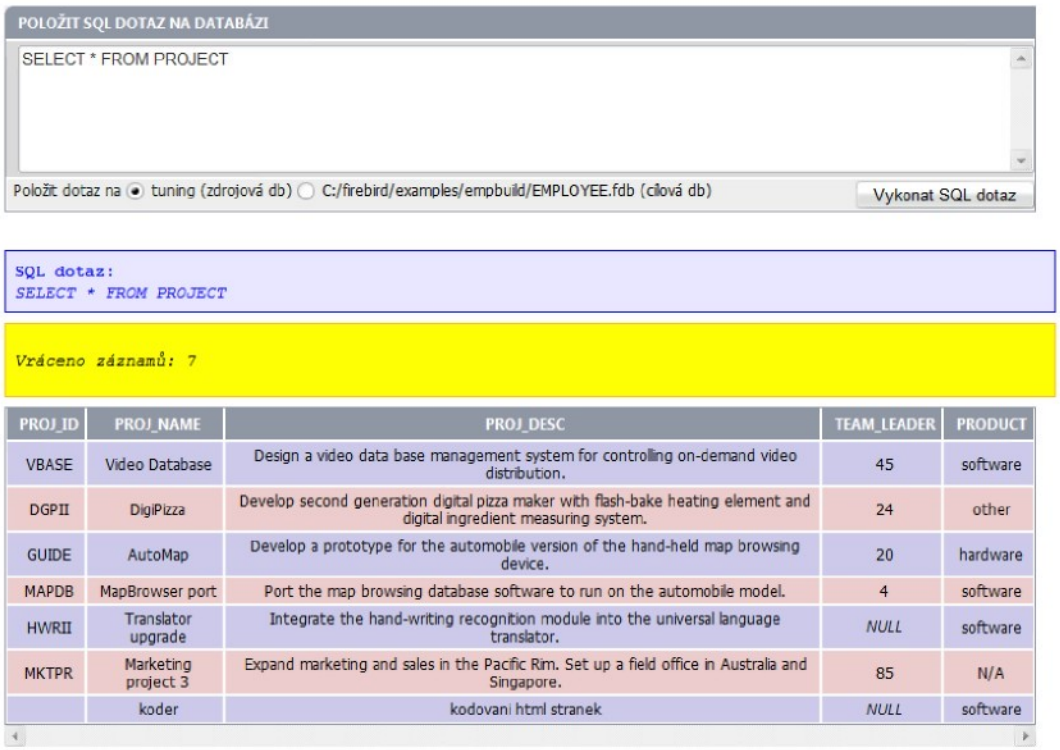
Po kliknutí na tlačítko *„Začít import dat“* dojde k importování dat, dle nastaveného namapování.

4.2.4. Formulář pro vykonání SQL dotazu

Součástí správy databází je i formulář (obrázek č. 14), do kterého je možné napsat jakýkoliv SQL dotaz a následně pak vykonat jak na cílové, tak i na zdrojové databázi.

obrázek č. 14 - formulář pro vykonání SQL dotazu

Jelikož není dotaz aplikací jakkoliv zkoumán a upravován, je nutné brát v potaz charakter dotazu, jenž bude vykonán (např. výběr všech záznamů z tabulky, kde jich je několik tisíc, by měl být omezen jen na několik desítek). Výsledek úspěšně provedeného dotazu je na obrázku č. 15.



PROJ_ID	PROJ_NAME	PROJ_DESC	TEAM_LEADER	PRODUCT
VBASE	Video Database	Design a video data base management system for controlling on-demand video distribution.	45	software
DGPII	DigiPizza	Develop second generation digital pizza maker with flash-bake heating element and digital ingredient measuring system.	24	other
GUIDE	AutoMap	Develop a prototype for the automobile version of the hand-held map browsing device.	20	hardware
MAPDB	MapBrowser port	Port the map browsing database software to run on the automobile model.	4	software
HWRII	Translator upgrade	Integrate the hand-writing recognition module into the universal language translator.	*NULL*	software
MKTPR	Marketing project 3	Expand marketing and sales in the Pacific Rim. Set up a field office in Australia and Singapore.	85	N/A
	koder	kodovani html stranek	*NULL*	software

obrázek č. 15 - výsledek provedeného dotazu

4.3. Kopírování Dat (převod dat)

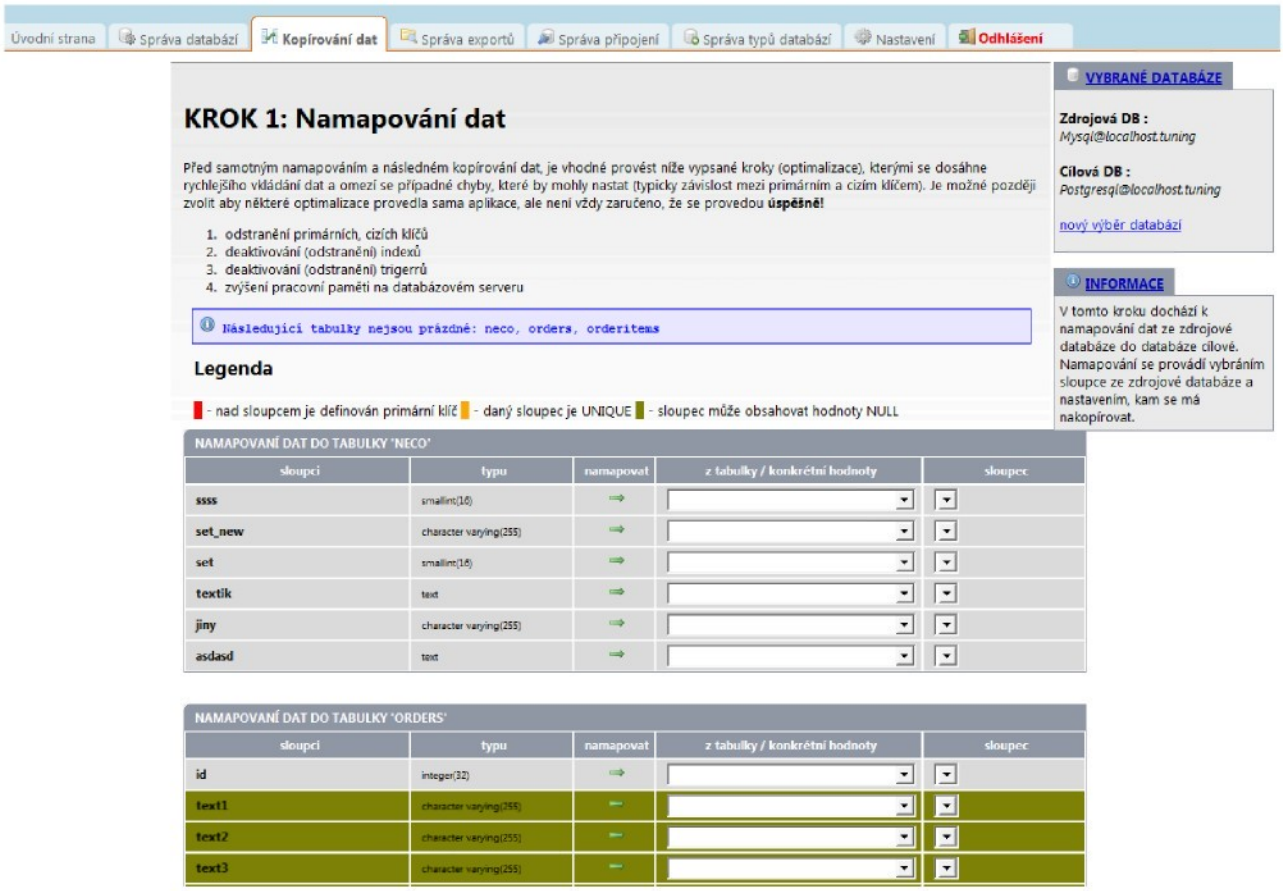
Před samotným kopírováním dat je nutné mít nastavené databáze, se kterými má aplikace pracovat (více informací v kapitole 5 - Výběr zdrojové a cílové databáze).

Převod dat probíhá ze zdrojové databáze do databáze cílové. V aplikaci je to realizováno ve čtyřech krocích:

1) Namapování dat
2) Kontrola namapování
3) Export dat
4) Import dat

4.3.1. Krok 1 – namapování dat

Před samotným mapováním dat je kontrolováno, zda jsou všechny tabulky v cílové databázi prázdné a v případě, že některé nejsou, je o tom uživatel patřičně informován (obrázek č. 16). Tato kontrola je v aplikaci z toho důvodu, že chceme ve většině případů kopírovat data do prázdných tabulek.



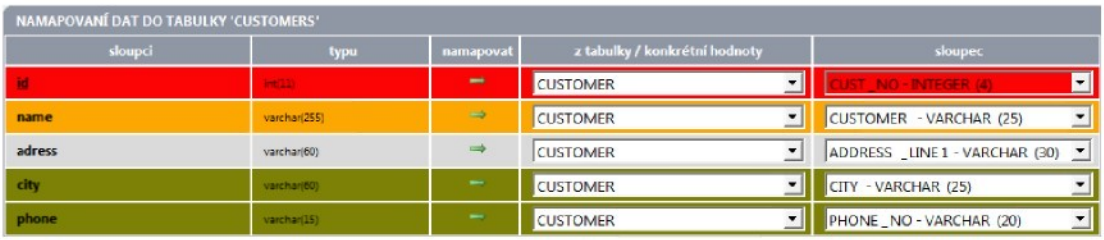
obrázek č. 16 – první krok

Kopírování dat by mělo probíhat do čistých tabulek. Na to se však nelze stoprocentně spolehnout, proto se na stránce nacházejí informace o optimalizacích, které by měly být na začátku provedeny. Je to i důvod proč aplikace ověřuje, zda jsou nad sloupci definovány primární klíče, unikátní indexy a zda mohou sloupce obsahovat hodnoty `NULL`. Tyto sloupce jsou pak barevně odlišeny od ostatních, což je nejjednodušší a nejpřehlednější způsob jak informovat uživatele. Ten tak na první pohled vidí, o jaké sloupce jde. Barvy byly voleny tak, aby byly jasně rozlišitelné. Součástí je i legenda, která informuje o tom, co která barva znamená. Ukázka je na obrázku č. 17.

obrázek č. 17 – barevné rozlišení sloupců

Na výše zobrazeném obrázku je možné vidět dvě tabulky z cílové databáze s vypsanými sloupci a informacemi o jejich typech (s maximální přípustnou délkou). Tyto informace se nachází v XHTML tabulkách, konkrétně v jejich levé části. Pravá část slouží k namapování dat.

Mapování probíhá výběrem tabulky ze zdrojové databáze a následným výběrem sloupce, kde sloupce jsou opět barevně rozlišovány. V některých případech pak uživatel vidí, že vybral sloupec se stejnými vlastnostmi (například unikátnost všech záznamů). Tyto vlastnosti jsou ovšem dostupné pouze v prohlížečích Opera a Internet Explorer. Ostatní vyhovující prohlížeče zobrazí barvu pouze při kliknutí na formulářový prvek `SELECT` a po následném vybrání sloupce nastaví barvu na bílou. Výsledek namapování jedné tabulky by pak mohl vypadat jako tomu je na obrázku č. 18.



obrázek č. 18 – namapovaná data do tabulky

Kromě možnosti výběru ze zdrojových tabulek a následně příslušných sloupců, lze také vybrat konkrétní hodnoty, které budou do sloupce vloženy. Příkladem může být přiřazení hodnoty `FALSE` sloupci určujícímu, zda je uživatel přihlášen. Vybírat lze z následujících možností:

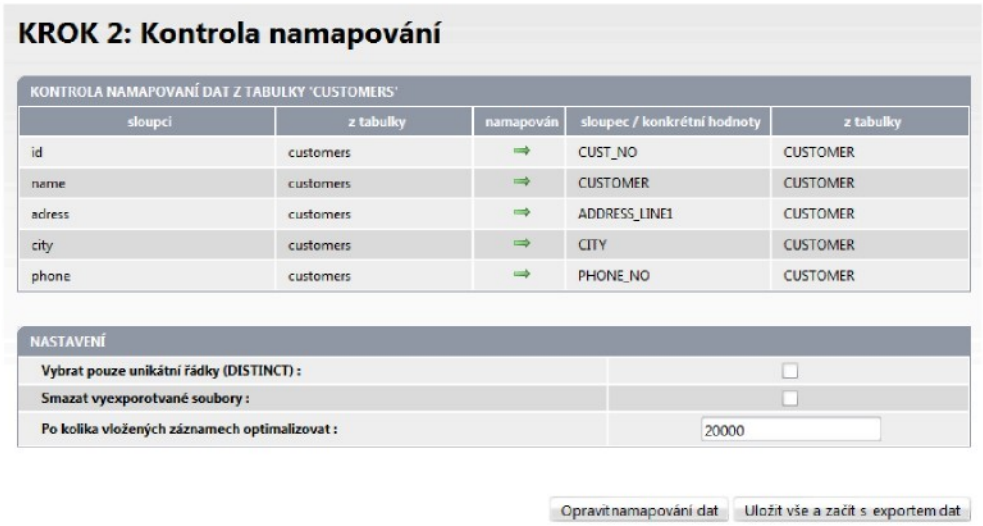
- **vložit DEFAULTní hodnoty** – vloží se do daného sloupce základní hodnoty, nadefinované při tvorbě struktury tabulky
- **vložit hodnoty NULL** – vloží se do daného sloupce hodnoty `NULL`
- **vložit hodnoty TRUE** – vloží se do daného sloupce hodnoty `TRUE`
- **vložit hodnoty FALSE** – vloží se do daného sloupce hodnoty `FALSE`
- **vložit aktuální časové/datové hodnoty** – vloží se do daného sloupce aktuální hodnota `TIMESTAMP`[1] formátovaná dle typu daného sloupce (např. hodnota `TIMESTAMP` se přeformátuje na typ sloupce `DATETIME`, tedy do podoby rok-měsíc-den hodina:minuta:sekunda)
- **vložit řetězec** – definuje se řetězec, který se vloží do daného sloupce
- **vložit číslo** – definuje se číslo, které se vloží do daného sloupce
- **Vložit unikátní ID (číslo)** – definuje se počáteční číslo, od kterého se bude začínat, poté dochází k postupné inkrementaci, vždy o jedničku

[1] počet sekund, které uběhly od 1 Ledna, 1970

4.3.2. Krok 2 – kontrola namapování

Po kliknutí na tlačítko „*Pokračovat*“ se aplikace přesune k druhému kroku. V něm dojde k rekapitulaci a následné kontrole namapování uživatelem.

Zobrazuje se také formulář s nastavením. V něm se nacházejí tři volby, z nichž poslední se týká optimalizací. Touto hodnotou se především určí, po kolika záznamech budou probíhat transakce a příkazy[2] pro optimalizaci.



obrázek č. 19 – kontrola namapování

4.3.3. Krok 3 – export dat

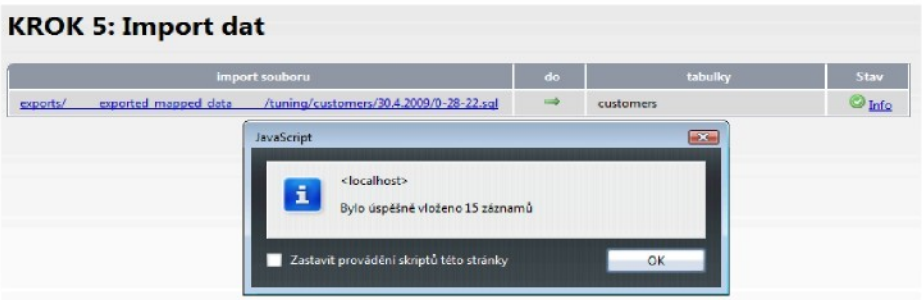
Po kliknutí na tlačítko „*Uložit vše a začít s exportem dat*“ se aplikace přesune ke třetímu kroku. Zobrazí se tabulka se seznamem cílových tabulek, pro které jsou exportována data s počtem záznamů a aktuálním stavem. V tomto kroku je pomocí AJAXu volán skript pro export dat, vždy pro jednu tabulku. Po jeho skončení je zavolán skript znovu, ale tentokrát pro jinou tabulku. Výsledek je na následujícím obrázku č. 20.

[2] například VACUUM pro PostgreSQL

obrázek č. 20 – úspěšně exportovaná data pro tabulku

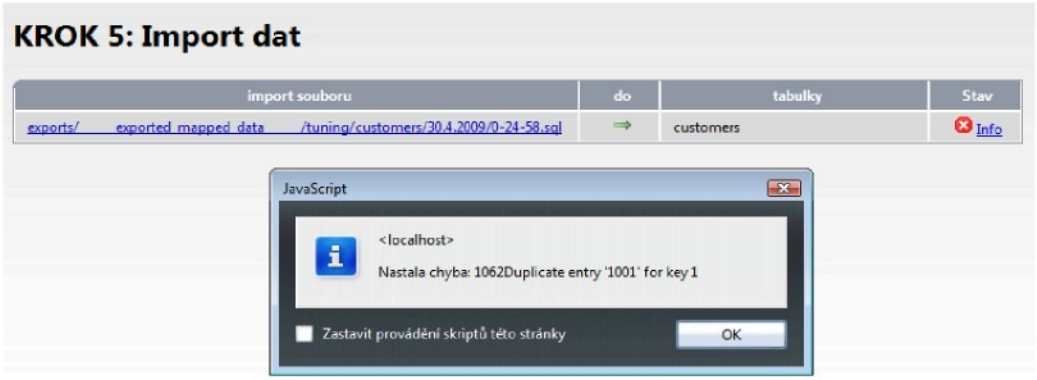
4.3.4. Krok 4 – import dat

Po kliknutí na tlačítko *„Pokračovat importováním vyexportovaných dat"* se aplikace přesune k poslednímu kroku. Zde je opět využito AJAXu a dochází k volání skriptu pro import dat, pouze pro jednu tabulku jako v případě exportu. Pokud je import proveden úspěšně, může si uživatel po kliknutí na odkaz v tabulce stáhnout vyexportovaný soubor a po kliknutí na tlačítko *„Info"* zjistí, kolik bylo úspěšně vloženo záznamů (obrázek č. 21).



obrázek č. 21 – úspěšně importována data do tabulky

V případě neúspěšného importu dat, se po kliknutí na tlačítko *„Info"* zobrazí informace o tom, k jaké chybě došlo (obrázek č. 22).



obrázek č. 22 – neúspěšné importování dat do tabulky

4.4. Správa exportů

V této části se nachází jednoduchý web souborový manažer, umožňující procházet pouze adresářovou strukturu složky (definována v nastavení aplikace) s exportovanými daty, stahovat, nahrávat a mazat soubory, mazat (prázdné i neprázdné) a vytvářet adresáře. Samotné procházení adresářů probíhá po kliknutí na příslušný název.

Manažer zobrazuje jak názvy adresářů a souborů, tak i informace jako datum vytvoření, práva, velikost a přípona souboru, popřípadě vlastníka a skupinu běží-li aplikace na Linuxu.

obrázek č. 23 - web souborový manažer

4.4.1. Vytvoření nového adresáře

Adresář lze snadno vytvořit zadáním jeho názvu a stisknutím tlačítka *„Vytvořit"*. Je ovšem potřeba dbát na to, že název může obsahovat pouze následující znaky `A-Za-z0-9_#@!&%-$.`

4.4.2. Nahrání souboru na server

Pro nahrání souboru na server stačí ve formuláři vybrat soubor a následně pak stisknout tlačítko „Nahrát soubor".

4.4.3. Odstranění souboru nebo adresáře

Odstranění souboru nebo adresáře je jednoduché. Stačí v tabulce ve sloupci *„Funkce"* kliknout na tlačítko pro odstranění (lze vidět na obrázku č. 22). Odstranit lze vždy pouze jeden soubor či adresář. U adresářů nezáleží na tom, zda je prázdný či nikoliv.

4.5. Správa připojení

Tato část slouží pro definování nebo upravení připojení k databázím. Rozdělena je na dvě části:

- **Správa hostitelských serverů**
- **Správa databází pro připojení**

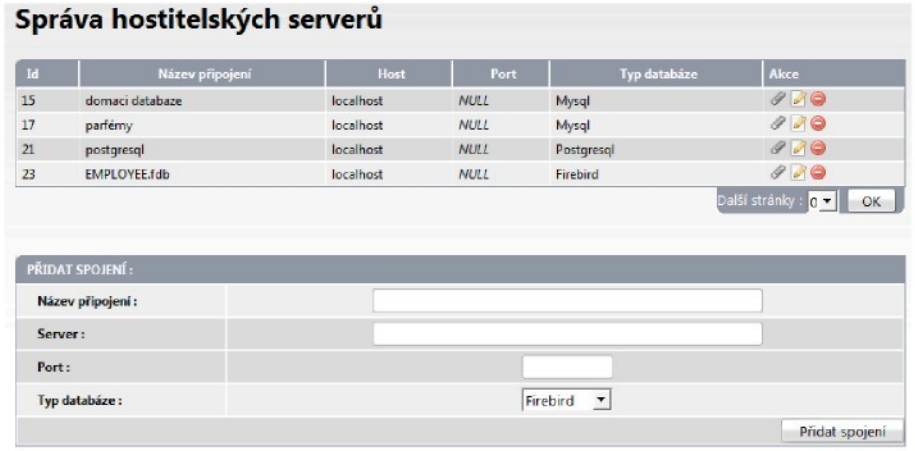
4.5.1. Správa hostitelských serverů

Zde se definují názvy pro připojení (alias), hostitelské servery, porty a typy databází (obrázek č. 24). Definovaná připojení jsou zobrazena v tabulce. V posledním sloupci *„Akce"* se nachází možné operace s připojením. Jedná se editaci a odstranění připojení a zobrazení přidružených databází.

Při přidávání nebo editaci připojení je ve formuláři nutné vyplnit vše, kromě položky *„Port"*, ta je volitelná. Pak již stačí kliknout na tlačítko *„Přidat připojení"* popřípadě *„Upravit spojení"*.

Popis jednotlivých položek formuláře:

- **Název připojení** – alias pro název serveru, na kterém běží databáze
- **Server** – název hostitelského počítače, na kterém běží databáze
- **Port** – číslo portu, na kterém se má aplikace připojit k hostitelskému serveru
- **Typ databáze** – o jaký typ databáze jde (MySQL, Firebird, PostgreSQL, Oracel, ...)



obrázek č. 24 – správa připojení k databázím

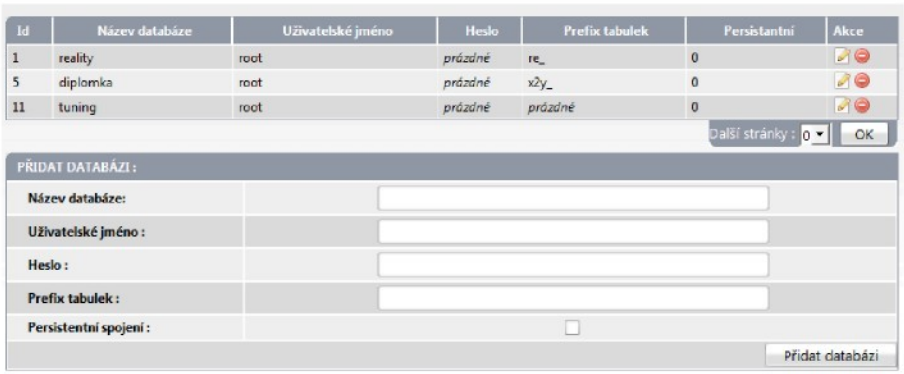
4.5.2. Správa databází pro připojení

Zde se definují přihlašovací údaje ke konkrétním databázím s možností definovat prefix tabulek (obrázek č. 25). Seznam přidružených databází se nachází v tabulce. V posledním sloupci *„Akce"* se nachází možné operace editace a odstranění databáze.

Při přidávání nebo editaci databáze je ve formuláři nutné vyplnit vše, kromě položek *Heslo*, *Prefix tabulek* a *Persistentní spojení*, ty jsou volitelné. Pak už stačí pouze kliknout na tlačítko *„Přidat databázi"* popřípadě *„Upravit databázi"*.

Popis jednotlivých položek formuláře:

- **Název databáze** – název databáze, kde jsou tabulky s daty pro aplikaci X2Y
- **Uživatelské jméno** – uživatelské jméno pro připojení k databázi
- **Heslo** – heslo pro připojení k databázi (volitelné)
- **Prefix** – prefix tabulek (volitelné)
- **Persistentní spojení** – zda se má aplikace připojovat k databázi při každém volání skriptů (volitelné)



obrázek č. 25 – správa databází pro připojení

4.6. Správa typů databází (správa ovladačů)

V této části aplikace se dá přidat, instalovat a odinstalovat podpora různých typů databází (správu ovladačů dostupných pro aplikaci X2Y).

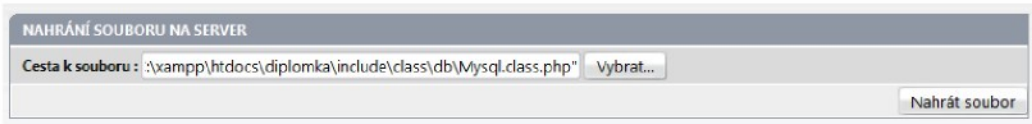
4.6.1. Přidání nového ovladače

Přidání nového ovladač je velice jednoduché. Stačí ve formuláři vybrat soubor s ovladačem a následně pak stisknout tlačítko „Nahrát soubor". Důležité je ovšem aby ovladač splňoval následující pravidla:

1) ovladač musí být jeden soubor
2) název souboru s ovladačem musí být typu

   `Typdatabaze.class.php` (tzn. Firebird.class.php, Mysql.class.php, Postgresql.class.php, ...)

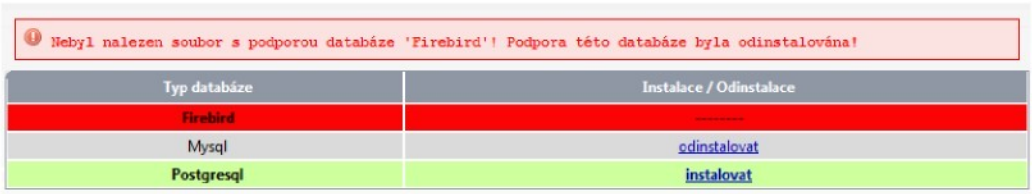
3) název potomka třídy `Database` musí být typ databáze začínající velkým písmenem, tzn. Firebird, Mysql, Postgresql, ...



obrázek č. 26 – přidání podpory nového typu databáze

4.6.2. Instalace a odinstalace ovladače

Na stránce se v tabulce zobrazuje seznam dostupných ovladačů dostupných pro aplikaci. Ukázka seznamu je na obrázku č. 27.



obrázek č. 27 - správa typů databází

Pokud je v seznamu typ, který jsme nově nahráli, popřípadě u kterého byla odinstalována podpora, je zvýrazněn zelenou barvou a nabízí se u něj možnost instalace po kliknutí na odkaz „instalovat".

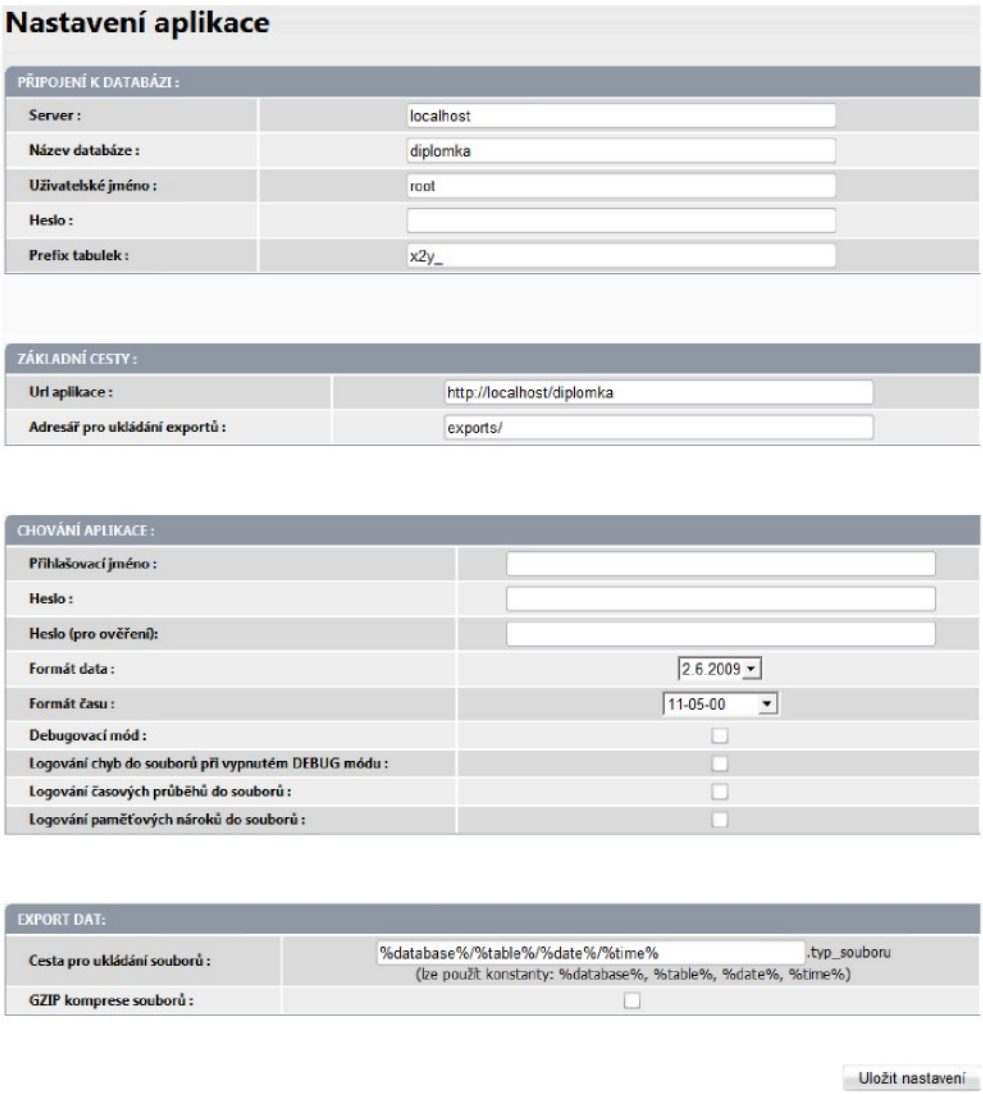
Pro odinstalování podpory jakéhokoliv typu databáze stačí kliknout na odkaz „odinstalovat". Po odinstalování, nebude možné daný typ zvolit jako zdrojovou nebo cílovou databázi. To znamená, že kdybychom například odebrali podporu databáze typu MySQL, nebude to mít vliv na funkčnost aplikace i přesto, že aplikace tuto databázi využívá. Znemožní se pouze možnost zvolit jako zdrojovou nebo cílovou databázi, právě databáze typu MySQL.

Pokud by byl nějaký ovladač nainstalován, ale soubor s ním nebyl nalezen, dojde k odinstalování, červenému zvýraznění a patřičnému informování uživatele jak je ukázáno na obrázku č. 27.

4.7. Nastavení

V této části se dá nastavit aplikace pomocí webového formuláře. Ten louží k nastavení základních údajů, které jsou pak při ukládání použity pro vytvoření nového souboru `config.php`. Tento soubor je možné editovat i ručně, ale doporučeno je to pouze zkušeným uživatelům, kteří vědí, co dělají. Nastavení je rozděleno do čtyř částí (viz obrázek č. 28):

- **Připojení k databázi**
- **Základní cesty**
- **Chování aplikace**
- **Export dat**



obrázek č. 28 - nastavení aplikace

4.7.1. Připojení k databázi

Zde se nachází informace nutné pro připojení k databázi, kterou využívá aplikace X2Y:

- **Server** – název serveru, kde běží MySQL databáze
- **Název databáze** – název databáze, kde jsou tabulky s daty pro aplikaci X2Y
- **Uživatelské jméno** – uživatelské jméno pro připojení k databázi
- **Heslo** – heslo pro připojení k databázi
- **Prefix** – prefix tabulek (defaultně x2y_)

4.7.2. Základní cesty

Zde se nastavují základní cesty:

- **Url aplikace** – základní url stránek (např. http://localhost/x2y/)
- **Adresář pro ukládání exportů** – adresář, do kterého se budou ukládat exportovaná data

4.7.3. Chování aplikace

Zde se nastavuje několik základních věcí ohledně chování aplikace:

- **Přihlašovací jméno** – přihlašovací jméno pro přihlášení do aplikace
- **Heslo** – heslo pro přihlášení do aplikace
- **Heslo (pro ověření)** – pro ověření, že bylo předchozí heslo zadáno správně
- **Formát data** – formát data, který bude používán v aplikaci (ovlivňuje proměnnou %date%, která může být použita při definování cesty pro ukládání souborů)
- **Formát času** – formát času, který bude používán v aplikaci (ovlivňuje proměnnou %time%, která může být použita při definování cesty pro ukládání souborů)

- **Debugovací mód** - zapne / vypne ladicí mód (více v kapitole 6 - LADICÍ MÓD)
- **Logování chyb do souborů při vypnutém DEBUG módu** - zapne / vypne logování chyb do souborů
- **Logování časových průběhů do souborů** - zapne / vypne logování časových průběhů do souborů
- **Logování paměťových nároků do souborů** - zapne / vypne logování paměťových nároků do souborů

4.7.4. Export dat

Zde se nastavují věci ohledně exportu dat:

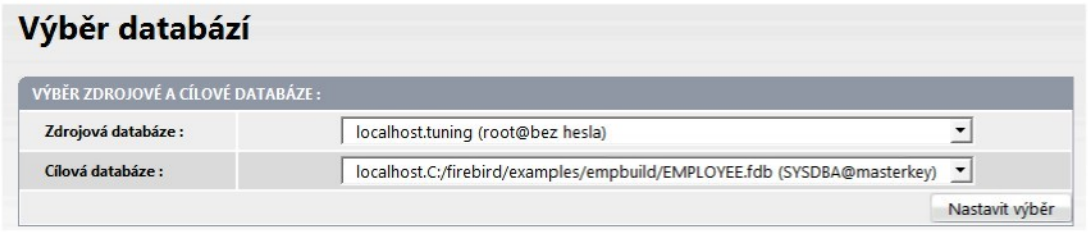
- **Cesta pro ukládání souborů** – struktura cesty pro ukládání souborů (např. nazev_databaze/nazev_tabulky/datum.typ_souboru) - lze použít proměnné: %database%, %table%, %date%, %time%, které nabývají vždy konkrétní hodnoty pro exportovaná data z tabulky
- **GZIP komprese souborů** – zapne / vypne GZIP kompresi souborů

5. VÝBĚR ZDROJOVÉ A CÍLOVÉ DATABÁZE

Stránka s formulářem (obrázek č. 29) pro výběr zdrojové a cílové databáze se zobrazí v případě, že doposud nebyly vybrány databáze, s kterými se bude pracovat a daná část aplikace to vyžaduje.

Pokud je potřeba nastavit databáze jiné, lze se na tuto stránku dostat po kliknutí na odkaz *„nový výběr databází“* v informačním okně *„VYBRANÉ DATABÁZE“*.

Po výběru databází a kliknutí na tlačítko *„Nastavit výběr“* se vše uloží a lze s vybranými databázemi v aplikaci pracovat.



obrázek č. 29 – výběr zdrojové a cílové databáze

6. LADICÍ MÓD

Pokud je v nastavení zaškrtnut *„Debugovací mód“* poběží aplikace v tzv. „debug módu“ (ladicí mód). Tento mód je v aplikaci z důvodu jejího testování. Po jeho zapnutí se ve spodní části (obrázek č. 30), pod patičkou stránek, zobrazí *„DEBUG WINDOW“* (ladicí okno), kde jsou k vidění základní systémové informace, vykonané SQL dotazy na databáze a případně chyby, které generovalo samotné PHP. Okno se dá schovat po kliknutí na odkaz *„DEBUG WINDOW“*.

obrázek č. 30 - ukázka zapnutého debug módu