Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Možnosti spolupráce MATLABu s databázemi

Tomáš Sedláček

Bakalářská práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Tomáš SEDLÁČEK

Studijní program: B2646 Informační technologie

Studijní obor: Informační technologie

Název tématu: Možnosti spolupráce MATLABu s databázemi

Zásady pro vypracování:

V teoretické části bude uvedena teorie databázových systému a shrnuty možnosti spolupráce MATLABu s databázemi. Praktickým výstupem bude propojení simulace s databází navrženou v MySQL.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Dušek F. Matlab a simulink - úvod do používání, 2. rozšířené vydání, Pardubice : Univerzita Pardubice, 2002. s 158. ISBN 80-7194-475-0. A další.

Vedoucí bakalářské práce: **Ing. Pavel Škrabánek**
Katedra řízení procesů

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlašuji:

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 30. 4. 2009

Tomáš Sedláček

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat vedoucímu práce Ing. Pavlu Škrabánkovi za cenné rady a připomínky, které mi poskytl během vypracování mé bakalářské práce.

SOUHRN

Tato práce se věnuje problematice propojení MATLABu s databázovými systémy. Konkrétně práce demonstruje možnosti spolupráce MATLABu s databázovým systémem MySQL. Výsledkem práce je souhrnný návod pro propojení MATLABu s databázemi i s balíkem skriptů, umožňujících propojit MATLAB a MySQL. Ty byly aplikovány při propojení simulace matematického modelu v programu SIMULINK s databází MySQL. V rámci práce rovněž bylo vytvořeno webové rozhraní, které zobrazuje aktuální vstupy a výstupy simulace a umožňuje řídit simulaci na dálku.

KLÍČOVÁ SLOVA

Databáze, MySQL, MATLAB,simulace, SIMULINK, Database Toolbox

TITLE

POSSIBILITIES OF COOPERATION MATLAB WITH DATABASES

ANOTATION

This work deals with problems of interconnection between MATLAB and database systems. Concretely, it demonstrates possibilities of the cooperation of MATLAB with MySQL database. A result of this work is a comprehensible manual for the interconnection of MATLAB with databases and also with a scripts package enabling interconnection between MATLAB and MySQL. They have been applied to the interconnection of mathematical model simulation in SIMULINK with MySQL database. Within the framework of this work, a web application has been created to display current simulation inputs and outputs and allow remote control of the simulation.

KEYWORDS

Database, MySQL, MATLAB, simulation, SIMULINK, Database Toolbox

Obsah

Seznam obrázků

1 Úvod

Tato práce se zabývá problematikou propojení relační databáze s prostředím MATLABu. Nástroj MATLAB je jedním z velkého množství programů na trhu umožňujících snadné řešení jednoduchých i složitých technických výpočtů. V některých případech však může nastat potřeba výměny dat mezi MATLABem a další aplikací. Jedním z prostředků pro výměnu dat může být některý z databázových systémů. Pro realizaci propojení MATLABu s databází je třeba mít nainstalovaný Database Toolbox, přičemž prostřednictvím jeho funkcí je možná spolupráce s databází. Tato spolupráce umožní propojit jakoukoliv simulaci s databází a je jedno, jestli tato simulace běží v MATLABu nebo v jeho nadstavbě SIMULINKu. Tohoto se dá lehce využít například při výuce v laboratořích, kdy reálný laboratorní model může být nahrazen matematickým modelem. Toto je také hlavní myšlenkou práce.

Vlastní práce spočívá ve vytvoření m-funkcí pro program MATLAB, které pracují s funkcemi Database Toolboxu a poskytnou načtení nebo vložení dat do databáze. Těchto m-funkcí se dále využívá k propojení realtimové simulace nelineárního matematického modelu v SIMULINKu. Tento matematický model je tvořen dvěma vstupy a výstupy. K ovládání modelu a vizualizaci časových průběhů vstupů a výstupů simulace byla vytvořena jednoduchá webová aplikace.

První část této práce je věnována teorii databázových systémů, konkrétně vytvoření přehledů, pojmů a definic z oblasti databázových systémů. Pro lepší pochopení dané problematiky jsou v textu uvedeny obrázky a praktické příklady.

Druhá část si dává za cíl osvětlit spolupráci MATLABu s databází s tím, že na začátku je krátké pojednání o nástrojích MATLAB a SIMULINK. Další část této kapitoly se věnuje Database Toolboxu, samotnému spojení a možnostem jeho využití při práci s databázovým systémem prostřednictvím jeho funkcí a grafického rozhraní.

Třetí část práce obsahuje praktické řešení problému při tvorbě databáze, m-funkcí, skriptů a webové aplikace.

2 Teorie databázových systémů

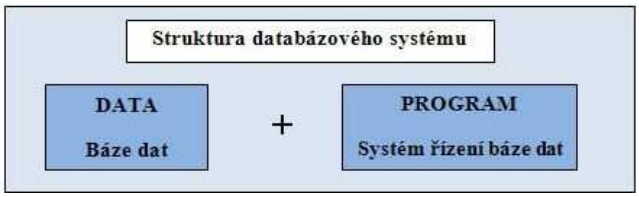
V této kapitole je stručně popsána teorie databázových systémů. Jsou zde vysvětleny základní pojmy, které jsou spjaté jak s tvorbou, tak i s používáním těchto databázových systémů.

2.1 Úvod do databází

Z pohledu dnešního světa je problematika databází dostatečně známá a snad každý člověk by dokázal okamžitě reagovat, když se řekne databáze. A nejen dnes, už dříve měli lidé potřebu někde uchovávat svá data a informace. Například jeskynní malby v pravěku byly také způsobem uchování „dat“. Postupem času se vznikem papíru, knihtisku a jiných technologií vznikaly tzv. kartotéky – dnes je ještě můžeme najít třeba u lékaře. Ty následně vystřídaly rejstříky, které odstraňovaly některé problémy kartoték, až k dnešním databázím, jak je známe my. Databáze jsou zkrátka velkým fenoménem a obklopují nás na každém kroku, aniž bychom si to uvědomovali. Za zmínku stojí využití databázových systémů pro evidenci občanů, automobilů, využití ve zdravotnictví, školství, dopravě, či dokonce ve vládních organizacích a bankách. A to není zdaleka konečný seznam oborů a odvětví, ve kterých se využívají databáze. Vývoj databází jde pořád kupředu a neustále vznikají nové technologie. Tyto technologie přispívají k tomu, že databázové systémy už neslouží jen jako skladiště dat. Moderní databázové systémy totiž obsahují mnoho funkcí pro řízení, optimalizaci, zálohování, monitorování atd. Pomocí těchto funkcí můžeme pohodlně a efektivně pracovat s databázovým systémem.

2.2 Databáze

Databáze je uspořádaná množina dat nebo informací, která je nějakým způsobem uložena na paměťovém médiu. Tato množina se souhrnně nazývá báze dat. Součástí databáze je však ještě systém, který slouží k manipulaci s daty a přístupu k nim. Tento systém se nazývá systém řízení báze dat (SŘBD). [1] Ukázku struktury databáze můžete vidět na obrázku 1.



Obrázek 1 - Ukázka struktury databázového systému

2.2.1 Báze dat

Jak už bylo řečeno, báze dat je nějaká uspořádaná množina dat. Tato data jsou organizována podle jednotných principů. Jedná se o datovou strukturu vysoké úrovně, jejíž jednotná organizace dat umožňuje jejich pohodlné sdílení. Zjednodušeně lze říci, že do báze dat vstupují nějaká data, která se pak prostřednictvím databázového systému stávají informacemi. Fyzické uložení báze dat je ve formě souboru na paměťovém médiu.[2]

2.2.2 Systém řízení báze dat

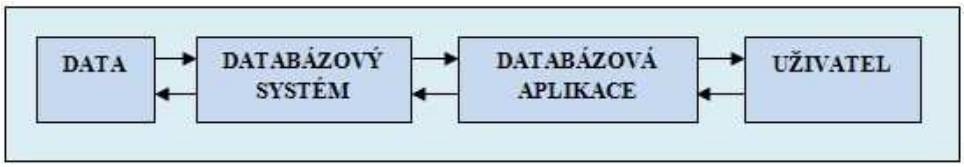
Systém řízení báze dat (databázový systém) představuje softwarové vybavení, které vytváří rozhraní mezi uloženými daty a uživatelem, případně aplikačním programem. Tento systém musí efektivně zpracovávat velké množství dat a být schopný řídit a definovat strukturu dat. Dnešní databázové systémy se mohou pochlubit řadou vlastností. Mezi tyto vlastnosti patří podpora definice datových modelů, využití jazyka vyšší úrovně pro manipulaci a definici dat, autentizaci uživatelů s použitím autorizace nad daty nebo správu transakcí. Databázový systém má také schopnost zotavit se a reagovat na chyby bez ztráty dat. [3]

Seznam některých databázových systémů:

- Oracle
- Firebird
- Microsoft Access
- Microsoft SQL Server
- Microsoft Visual FoxPro
- MySQL

2.2.3 Databázová aplikace

Jedná se vlastně o aplikaci, která komunikuje se systémem řízení báze dat. Jejím účelem je zprostředkovat uživateli přístup k datům pomocí formulářů, kde uživatelé zadávají data. Tato data se pak posílají databázovému systému, který je následně uloží do databáze. Stejným způsobem, ale v opačném pořadí, pak probíhá poskytnutí dat z databáze. Pro lepší pochopení této komunikace je zde uveden obrázek 2.



Obrázek 2 - Komunikace mezi uživatelem a databází

2.3 Databázové modely

Databázové modely můžeme rozdělit z hlediska ukládání, reprezentace a vztahů mezi daty. Pro úplnost je zde uveden přehled databázových modelů.

Přehled databázových modelů:

- **Hierarchický databázový model** – data jsou uspořádána hierarchicky (stromová struktura) a jsou reprezentována rodičem a potomkem.
- **Síťový databázový model** – jedná se o jakési vylepšení hierarchické struktury. Záznamy jsou reprezentovány uzly, které jsou mezi sebou ve vztahu vlastník – člen.
- **Objektově orientovaný databázový model –** tento model vznikl příchodem objektově orientovaného programování (OOP). Data jsou zde sdružována do objektů.
- **Relační databázový model** – relační databázový model je dnes ze všech modelů používán nejčastěji. Sdružuje data do tzv. relací. Protože se jedná o nejrozšířenější model, bude mu věnována následující kapitola.

2.4 Relační databázový model

Vznik relačního modelu se datuje k roku 1969 a stojí za ním E. F. Codd. Jeho představou bylo založit databáze na teorii množin a predikátové logice. Název relač-

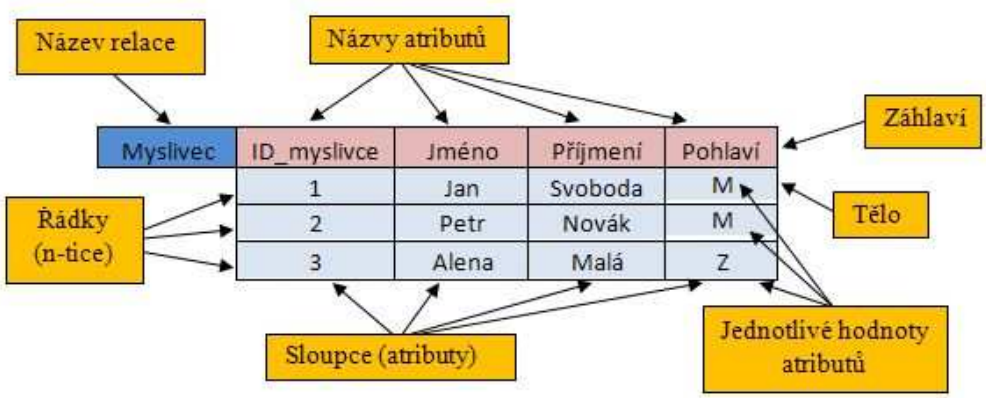
ní databáze vznikl odvozením pojmu relace z teorie množin. Relace má zavedený pomocný aparát, který se nazývá schéma relace. Tento aparát říká, jaký je název relace, kolik obsahuje sloupců, jak se tyto sloupce nazývají a jaké jsou množiny přípustných hodnot pro daný sloupec. Tyto přípustné hodnoty se nazývají domény. [4]

2.4.1 Definice relace

„Z matematického hlediska je relace R podmnožina kartézského součinu domén (množin přípustných hodnot) D1, D2,…,Dn, které přísluší jednotlivým atributům A1, A2,…, An, tj. R ⊆ D1 ×…× Dn. Prvky relace jsou uspořádané n-tice (r[A1], r[A2],…,r[An]) hodnot náležejícím doménám D1, D2,…Dn.“ [2]

2.4.2 Tabulka

Relační databázový model sjednocuje data do tzv. relací, které nazýváme tabulky.[4] Tato data jsou pak uložena v jednotlivých n-ticích, které představují řádky tabulky. Tabulka je tedy složena z několika řádků a sloupců, přičemž každý sloupec musí mít jedinečný název a požadovaný datový typ pro reprezentaci jednotlivých dat. Řádky pak obsahují jednotlivé hodnoty a platí pro ně, že jeden řádek se rovná jednomu záznamu. Každý řádek by měl mít také nějaký jednoznačný identifikátor, aby nedocházelo k duplicitě záznamů v tabulce. Ukázku relace databázové tabulky je možné vidět na obrázku 3, kde je uvedena jako seznam myslivců.



Obrázek 3 - Relace databázové tabulky

2.4.3 Integrita relační databáze

Integrita databáze zajišťuje konzistenci dat uložených v databázi. Proto, abychom mohli používat databázi jako zdroj dat, je nutné zajistit, aby se do databáze

dostaly jen ty množiny dat, které tam patří. Je tedy nutné mít nějaké mechanismy, které by to zabezpečovaly. Tyto mechanismy můžeme nazvat jako Integritní omezení.[5]

Data jsou tedy konzistentní, pokud vyhovují Integritním omezením. Znamená to, že při úpravě nedošlo k poškození či ztrátě dat, nebo že v databázi nejsou data, která tam nepatří.[5] Příkladem může být seznam posedů, které vlastní smazaný uživatel (myslivec).

Pro zabránění těmto situacím se používají právě integritní omezení, která zajistí například při smazání daného uživatele (myslivce), že se odstraní i všechny jeho záznamy v ostatních tabulkách.

Druhy integritních omezení:

- **Entitní –** jedná se o povinné integritní omezení a o primární klíč tabulky.[5] Primární klíč (PK)[1] je jednoznačný identifikátor řádku tabulky. V tabulce je definovaný vždy pouze jeden primární klíč a pole tohoto klíče nesmí obsahovat hodnotu NULL.
- **Doménové –** tento druh integritního omezení pracuje na úrovni sloupců tabulky a definuje omezení na datový typ nebo rozsah hodnot. [5]
- **Referenční –** tato integrita se zabývá vztahy mezi dvěma tabulkami. Referenční integrita se definuje cizím klíčem (FK)[2] a tabulky, které se účastní tohoto vztahu, se nazývají nadřízená (master) a podřízená (slave). Pravidlo referenční integrity říká, že pro každý záznam v podřízené tabulce musí existovat záznam se stejným klíčem v nadřízené tabulce. [5]
- **Aktivní referenční integrita –** definuje, jak se má databázový stroj zachovat při porušení referenční integrity.

2.4.4 Normalizace relační databáze

Cílem normalizace databáze je zjednodušení a optimalizace databázových tabulek tak, aby byla databáze přehlednější, rozšiřitelnější a výkonnější. Je všeobecně známo, že čím je tabulka ve vyšší normální formě, tím kvalitněji je navržena a práce s ní je pak o něco snazší.

[1] Primary Key
[2] Foreign Key

Při normalizaci vlastně dochází k odstranění opakujících[3] se dat, k omezení složitosti tabulky, přičemž některé složité relace rozdělíme do více tabulek, a také k zabránění tzv. aktualizačních anomálií. To znamená, abychom po vymazání všech knih daného autora nepřišli o data, která se týkají samotného autora.[6]

Normalizované schéma musí zachovat všechny závislosti, stejně jako relace musí zachovat původní data. Z toho vyplývá, že pomocí přirozeného spojení musíme dostat původní data.[6] Normální formy mají několik stupňů a tyto stupně zde budou uvedeny.

Normální formy:

- **0. normální forma** – tabulka je v této normální formě, pokud existuje alespoň jedno pole, které obsahuje více než jednu hodnotu.
- **1. normální forma –** jedná se o nejjednodušší formu, která nám říká, že lze dosadit do každého pole pouze jednoduchý datový typ. Atributy obsahují pouze atomické (nedělitelné) hodnoty.
- **2. normální forma –** tabulka je v druhé normální formě, pokud je v první normální formě a přitom platí, že každý neklíčový atribut je plně závislý na primárním klíči jako celku, nejen na jeho části.
- **3. normální forma –** tato normální forma je splněna za předpokladu, že jsou splněny dvě předchozí normální formy a jestliže každý neklíčový atribut není tranzitivně závislý na žádném klíči.
- **BCNF[4] -** „*Schéma relace R(A:D) je v Boyce-Coddově normální formě, je-li v první normální formě a platí-li pro každou funkční závislost* $X \rightarrow A$*, která není triviální, že X je klíčem v R a A je neklíčový atribut.*“[2]
- **4. normální forma –** tabulka je ve čtvrté normální formě, je-li ve třetí nebo Boyce-Coddově normální formě a popisuje nějakou příčinnou souvislost nebo jeden fakt.
- **5. normální forma –** tabulka je v páté normální formě, pokud je ve čtvrté normální formě a není možné do ní přidat nový atribut nebo skupinu atributů tak, aby se vlivem skrytých závislostí rozpadla do několika dílčích tabulek.[6]

[3] Někdy též redundantních
[4] Boyce-Coddova normální forma

Normalizace databázových tabulek je určitě vhodná při velkém množství dat nebo velké složitosti databáze. V praxi se většinou provádí normalizace do třetí normální formy, protože čtvrtá a pátá normální forma už jsou poněkud složitější a vyžadují vyšší znalosti z oblasti návrhu databáze.

2.5 Konceptuální modelování

Konceptuálního modelování se využívá v první fázi návrhu databáze, kdy se komunikuje se zákazníkem nebo uživatelem dané databáze a dochází ke shromažďování a formulaci požadavků, co bude vlastně v databázi uloženo. Z těchto informací se vychází a vytvoří se konceptuální model.

„*Konceptuální modely jsou pokusem umožnit vytvoření popisu dat v databázi, tj. konceptuálního schématu, nezávisle na fyzickém uložení databáze. Tento popis by měl nejvěrněji vystihovat konceptuální pohled člověka na danou část reálného světa*.“ [7]

Zjednodušeně řečeno, konceptuální model by měl co nejvěrohodněji zachycovat pohled člověka na danou část reálného světa.

2.5.1 Základní pojmy konceptuálního modelování

Entita

Entity představují nějaké objekty reálného světa, které existují nezávisle a jsou jednoznačně rozlišitelné od ostatních objektů. Vzájemně podobné entity se potom sdružují do skupin, které představují určitý typ. Takové skupiny potom nazýváme jako entitní množiny. [2]

Entitami, jak už zde bylo zmíněno, mohou být jak objekty reálného světa, např. myslivec Petr Novák, ale také abstraktní objekty typu - posed nebo krmelec, které představují druh mysliveckých staveb. Entitní množinou, kterou tvoří entity stejného typu, může být jak množina entit myslivci {Petr Novák, Jan Svoboda, Alena Malá}, tak množina entit myslivecké stavby {posed, krmelec, kazatelna}.

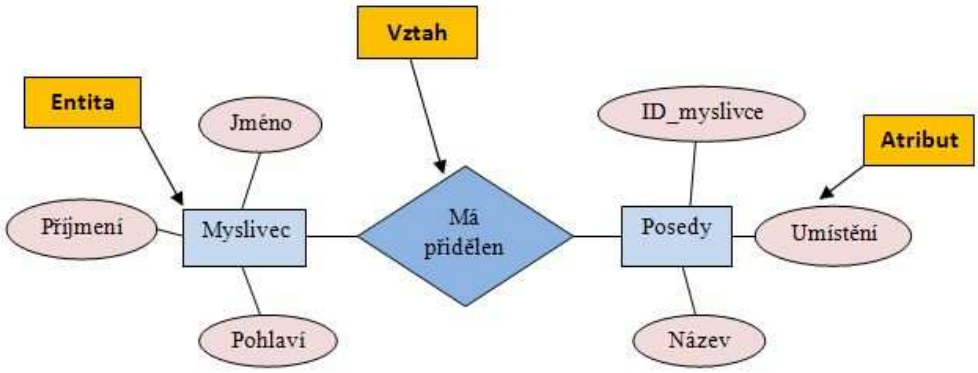
Vztah

Vztah si můžeme představit jako druh jakési vazby mezi dvěma nebo více entitami. Příkladem může být např. entita {myslivec Petr Novák, pohlaví muž}, která může být ve vztahu {má přidělen} k entitě {stavba posed, č.1}.

Atribut

Pod atributem rozumíme funkci, která přiřazuje entitám nebo vztahům nějakou hodnotu. Tato hodnota určuje vlastnost dané entity nebo vztahu. [7] Jako příklad lze uvést datum konání honu, na který jsou přihlášeni jednotliví myslivci, nebo jméno a příjmení daného myslivce, apod.

Ukázku základních pojmů v praxi popisuje obrázek 4.



Obrázek 4 - Ilustrační ukázka jednotlivých pojmů

2.5.2 E-R model

E-R[5] model patří k nejznámějším konceptuálním modelům a pracuje se základními prvky, kterými jsou entitní typy, jejich atributy a vzájemné vztahy. Model E-R byl poprvé uveden Peterem Chenem v roce 1976 a postupem času docházelo k jeho rozšiřování, až se z něho stal uznávaný standard. [2]

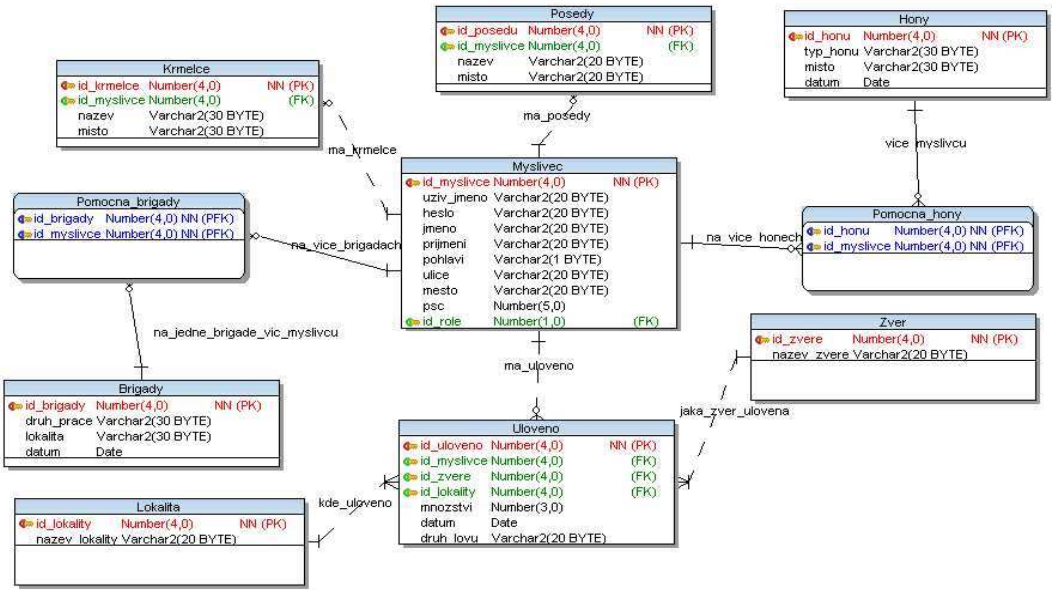
„E-R model je množina pojmů, které nám pomáhají na konceptuální úrovni abstrakce popsat uživatelskou aplikaci s účelem následně specifikovat strukturu databáze.“ [7]

Každá entita by měla být jednoznačně identifikovatelná. Proto používáme identifikační klíč, který je vlastně atribut, jehož hodnota slouží k identifikaci kon-

[5] Entity-Relationship

krétní entity. [7] Entitní typy v E-R modelu můžeme rozdělit na silné a slabé. Jako silné entity si můžeme představit takové entity, které jsou identifikovány pouze vlastními atributy, zatímco slabé entity jsou identifikačně závislé na jiném entitním typu. [2] Můžeme se také setkat s pojmem závislá entita, jejíž existence je závislá na jedné nebo více entitách.

Při tvorbě E-R modelů využíváme určitých grafických objektů, které nahrazují jednotlivé pojmy. Entity se v těchto modelech popisují pomocí obdélníků, vztahy se znázorňují prostřednictvím kosočtverců a atributy používají elipsu. Hrany ukazují, které entitní typy jsou zapojeny do jednotlivých vztahů. E-R model je nejčastěji používaný model pro abstraktní modelování v technologii CASE a jeho výhodou je, že není vázán na určitý datový model. To nám umožňuje modelování světa nezávisle na datových strukturách a jejich počítačové reprezentaci. [2] Příklad moderního E-R diagramu je uveden na následujícím obrázku 5.



Obrázek 5 - Ukázka E-R diagramu, vytvořeného v prostředí TDM3[6]

2.5.3 Vztahy mezi tabulkami

Vztahy, nebo též relace mezi tabulkami nám slouží ke spojení dat, která spolu nějakým způsobem souvisí. Tato data jsou pak uložena v jednotlivých tabulkách, které vstupují do vztahu. [8] Tabulky, svázané určitým vztahem, můžeme souhrnně nazvat jako účastníci vztahu a počet účastníků vztahu můžeme vyjádřit jako stupeň

[6] Toad Data Modeler 3 – nástroj pro modelování databází

vztahu. Dle tohoto stupně lze určit, zda se jedná o binární vztah (vztah mezi dvěma tabulkami) nebo unární vztah (tabulka je spojena sama se sebou).

V případě existence vztahů mezi tabulkami můžeme definovat dvě základní vlastnosti, které se nazývají kardinalita a parcialita.

Kardinalita

Kardinalita vztahu nám říká, kolik řádků jedné tabulky může vstoupit do vztahu z kolika řádky druhé tabulky. Existují tři typy kardinálního binárního vztahu a tyto vztahy zde budou představeny.

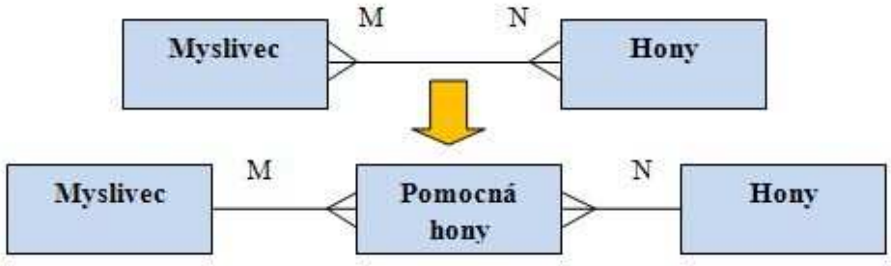
- **Vztah 1:1 –** tento vztah vyjadřuje situaci, kdy jednomu řádku v jedné tabulce odpovídá právě jeden řádek ve druhé tabulce. Jedná se o nejjednodušší typ vztahu, který se však používá jen ojediněle, a většinou se takové záznamy ukládají do jediné tabulky. [8] Jako příklad je možno uvést, že jeden manžel má jednu manželku nebo auto v jednu chvíli řídí jeden řidič a zároveň jedno auto je řízeno právě jedním řidičem.

Obrázek 6 - Ukázka vztahu 1:1

- **Vztah 1:N** – pro tento vztah platí, že jednomu řádku z jedné tabulky může odpovídat více řádků z druhé tabulky. Jedná se o nejpoužívanější typ vztahu, který odpovídá mnoha situacím v reálném životě. [8] Pro ilustraci je možno uvést příklad, kdy jeden myslivec může vlastnit více posedů, a zároveň jeden posed je vlastněn právě jedním myslivcem nebo jeden zákazník nemusí mít žádnou nebo může mít více objednávek a právě jedna objednávka musí mít jednoho konkrétného zákazníka. V tomto případě je nutná existence zákazníka, protože objednávka bez zákazníka existovat nemůže.

Obrázek 7 - Ukázka vztahu 1:N

- **Vztah M:N –** tento vztah umožňuje situaci, kdy několika záznamům jedné tabulky odpovídá několik záznamů druhé tabulky. Co se týče relačních databází, tak nelze uskutečnit tento vztah přímo, ale s využitím pomocné tabulky. Tato pomocná tabulka má k oběma tabulkám vztah jedna k více a obsahuje jako cizí klíče primární klíče obou tabulek. Jako příklad je možno uvést, že myslivec může navštívit více honů, a zároveň každý hon čítá více myslivců nebo jeden zákazník si může koupit více výrobků, a zároveň každý výrobek si může koupit několik zákazníků.



Obrázek 8 - Ukázka vztahu M:N s použitím pomocné tabulky

Unární vztahy

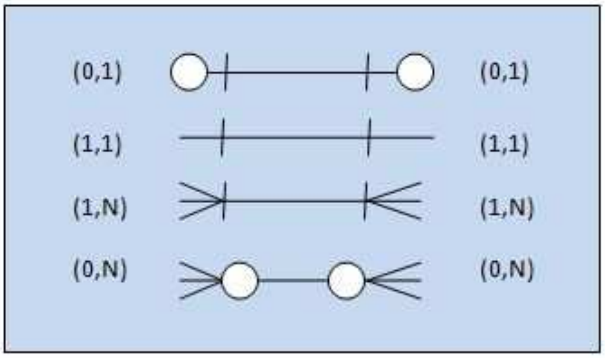
Unární vztahy mají zvláštnost v tom, že na rozdíl od binárních vztahů je tabulka napojena sama na sebe. Modelují se stejným způsobem jako binární vztahy s tím rozdílem, že primární a cizí klíč se odkazují na stejnou tabulku. Typickým příkladem může být vztah mezi Zaměstnancem a Nadřízeným, kdy nadřízený je vlastně také zaměstnanec a může mít i další nadřízené. [9] Příklad takové tabulky je uveden na následujícím obrázku 9, kde je zobrazen vztah mezi zaměstnancem a nadřízeným.

Obrázek 9 - Ukázka unárního spojení tabulky

Parcialita

Parcialita vyjadřuje povinnost nebo nepovinnost existence role příslušné entity vztahu. Vztah může být vyjádřen jako jednostranně nebo oboustranně parciální. Jednostranně parciální vztah znamená, že například zaměstnanec musí náležet jedné

pojišťovně, a zároveň pojišťovna nemusí mít v evidenci ani jednoho zaměstnance. Oboustranně parciální vztah říká, že zaměstnanec nemusí náležet k žádné pojišťovně a pojišťovna nemusí evidovat ani jednoho zaměstnance. [9] Přehledné vyznačení parciality společně s kardinalitou je zobrazeno na následujícím obrázku 10.



Obrázek 10 - Ukázka vyznačení parciality společně s kardinalitou

3 Možnosti spolupráce MATLABu s databázemi

V předešlé kapitole, která se zabývala teorií databázových systémů, jsou vysvětleny některé základní principy a pojmy z oblasti databázových systémů. Následující kapitola se zabývá programem MATLAB a jeho propojením s databázovými systémy.

3.1 Základní popis prostředí MATLABu

Ještě než se začneme zabývat samotným spojením MATLABu s databázemi, je nutné si vysvětlit, co vůbec MATLAB je, k čemu ho můžeme použít a jaké funkce nabízí.

MATLAB, stejně jako např. MathCad nebo Mathematica, patří do skupiny matematických programů, které se zabývají numerickými výpočty. Jeho název je odvozen z anglického matrix laboratory a byl vytvořen firmou MathSoft Inc., aby poskytoval jednoduchý přístup k matematickým funkcím a knihovnám. Původně byl určen pro UNIX, ale dnes ho již můžeme pohodlně využívat i v prostředí Windows.

Slovy firemní literatury „*MATLAB je vysoce výkonný jazyk pro technické výpočty. Integruje výpočty, vizualizaci a programování do jednoduše použitelného prostředí, kde problémy i řešení jsou vyjádřeny v přirozeném stavu.*“ [10] Jinak řečeno, jde o interaktivní systém, který má jako základní datový typ dvourozměrné nebo vícerozměrné pole, bez nutnosti deklarovat rozměry. Tato vlastnost, společně se zabudovanými funkcemi, vede k rychlejšímu řešení různých technických problémů[7], než řešení problému v klasickém programovacím jazyce typu C nebo Fortran. [10]

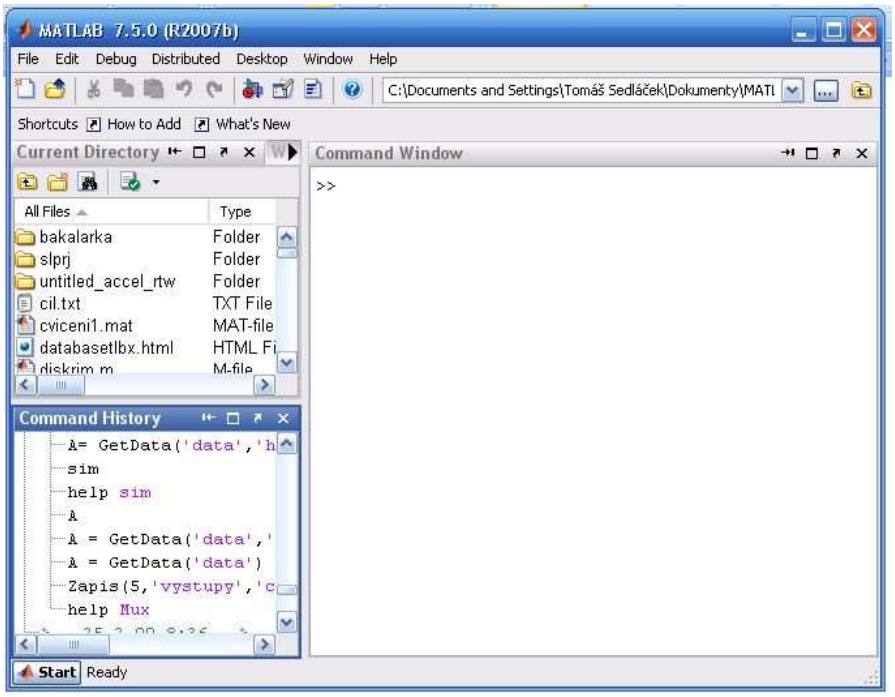
Některé oblasti použití: [10]

- inženýrské výpočty
- vývoj algoritmů
- modelování, simulace a vývoj prototypů
- analýza dat a jejich vizualizace
- inženýrská grafika
- vývoj aplikací včetně GUI

[7] týkajících se vektorů a matic

MATLAB je také s oblibou používán na školách k výuce matematiky, ale i jiných předmětů. A nejen ve školství, MATLAB nachází využití také v průmyslu pro vývoj a analýzu dat. Z důvodu jeho jednoduchého rozhraní, které je tvořeno příkazovou řádkou, lze za velmi krátkou dobu dosáhnout dobrých výsledků. Jako příklad lze uvést výpočet diferenciálních rovnic nebo polynomů, jejichž výpočet by za normálních okolností byl značně složitý. MATLABu by tyto výpočty trvaly jen několik sekund, což z něj dělá mocný nástroj, jehož obliba stále roste.

Pracovní prostředí MATLABu, tedy v jeho základní podobě, je tvořeno třemi okny viz obrázek 11, z nichž nejdůležitějším je asi okno s příkazovým řádkem. Zbylá dvě okna slouží pro zjednodušení práce se soubory a ukládání historie příkazů.

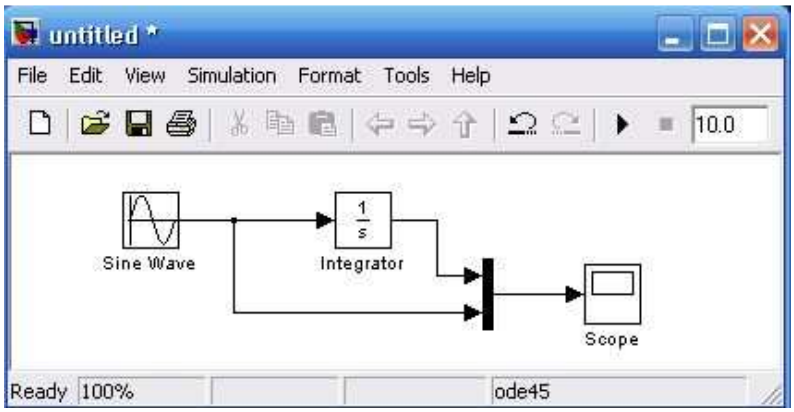


Obrázek 11 - Uspořádání oken v MATLABu

3.1.2 SIMULINK

MATLAB obsahuje jako samostatnou nadstavbu SIMULINK, což je vlastně program, který slouží pro simulaci a modelování dynamických systémů. Pro simulaci těchto systémů se používají algoritmy pro řešení numerických nelineárních diferenciálních rovnic. Lze pomocí něj tedy získat časové průběhy výstupních veličin v závislosti na časovém průběhu vstupních veličin a počátečních podmínkách. [10]

SIMULINK využívá podobně jako MATLAB grafické rozhraní, které se však liší samotným ovládáním a prací v něm. Jak již bylo řečeno, MATLAB se z převážné části ovládá z příkazové řádky, přičemž při práci se SIMULINKem využíváme grafický editor a knihovny bloků. Tyto bloky lze pak jednoduše přesunout do editoru, vhodně propojit a vytvořit z nich model, s kterým pak lze provádět simulační výpočty. Pro ilustraci je zde uveden obrázek 12, na kterém je ukázka jednoduchého modelu.



Obrázek 12 - Jednoduchý model zobrazující dva sinusové signály

SIMULINK tedy umožňuje uživateli snadnou a efektivní práci při tvorbě lineárních, nelineárních, časově závislých nebo spojitých systémů ve velmi krátké časové době, pouhým přesouváním funkčních bloků.

3.1.3 Toolboxy

MATLAB obsahuje kromě základních vestavěných funkcí také knihovny funkcí, které se nazývají toolboxy. Tyto knihovny rozšiřují použití MATLABu v různých odborných nebo technických odvětvích.

Toolboxy jsou napsané v jazyce MATLAB a nabízejí speciální funkce pro řešení problému daného odvětví. Využívají systém Handle Graphics, který je založen na postupech, jenž jsou pro uživatele jednodušší, to znamená, že umožňují uživateli zadat data, vybrat metodu a spustit výpočet. [10] Zkrátka můžeme používat toolboxy pro řešení nějakého problému, aniž bychom měli nějaké znalosti o použitém výpočtu.

Toolboxy se instalují zároveň s programem MATLAB s tím, že je nutné nakoupit jejich potřebné licence. Toolboxů je celá řada a jak již bylo řečeno, liší se použitím v daném odvětví, proto jako příklad budou uvedeny pouze některé. Za zmínku

stojí třeba Symbolic Math Toolbox pro symbolické výpočty, Real Time Toolbox, který zajišťuje práci v reálném čase, dále třeba Statistic Toolbox nebo Financial Toolbox pro statistické a bankovní výpočty. Asi nejdůležitějším toolboxem, alespoň co se týká této práce, je Database Toolbox, který zajišťuje spojení MATLABu s databázemi. Tento toolbox bude podrobně vysvětlen v následující kapitole.

3.2 Propojení MATLABu a databází

Až doposud bylo popisováno prostředí MATLABu a problematika databázových systémů. Nyní se tedy budeme zabývat samotným spojením MATLABu s databázemi.

V minulé kapitole věnované MATLABu bylo napsáno, že v tomto prostředí existují knihovny funkcí, které se nazývají toolboxy. MATLAB má ve své bohaté nabídce toolboxů tzv. Database Toolbox, který byl vytvořen k tomu, aby dokázal propojit MATLAB s databázemi a poskytnul základní funkce nutné pro komunikaci a manipulaci s databázovým systémem. Proto je potřebné mít tento toolbox nainstalován a připraven k použití.

3.2.1 Database Toolbox

Database Toolbox je jedním z rozsáhlé kolekce toolboxů, které jsou k dispozici v MATLABu, a umožňuje používat jeho funkce pro import a export dat mezi MATLABem a relační databází. Pro ilustraci lze uvést příklad, kdy potřebujeme svá data výpočetně nebo analyticky zpracovat a chceme použít prostředí MATLABu. Pomocí tohoto toolboxu jednoduše načteme data z databáze, zpracujeme je a následně uložíme zpět do databáze.

Pro používání Database Toolboxu je potřebná znalost základů MATLABu, především při tvorbě matic a struktur. Uživatel by měl být rovněž seznámen s pojmy používanými v souvislosti s databázemi, jako jsou tabulka, atributy nebo dotazy.

Součástí Database Toolboxu je nástroj Visual Query Builder, který poskytuje uživateli základní grafické rozhraní, pomocí kterého lze pohodlně pracovat s databází. Použití VQB je jedna ze dvou variant, jak lze pracovat s Database Toolboxem. Druhá varianta spočívá přímo v použití funkcí Database Toolboxu, přičemž je nutná jejich znalost

Aby se Database Toolbox dal opravdu použít, je třeba si vybrat některý databázový systém a nainstalovat ho, protože MATLAB nebo konkrétně Database Toolbox sám o sobě neobsahuje žádný nástroj pro správu databázových systémů. Přehled použitelných databázových systémů zobrazuje následující seznam.

Databáze podporované Database Toolboxem: [11]

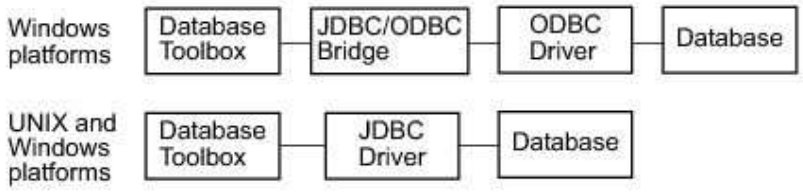
- IBM DB2®
- IBM Informix®
- Ingres®
- Microsoft Access®
- Microsoft Excel®
- Microsoft SQL Server®
- MySQL®
- Oracle®
- PostgreSQL®
- Sybase SQL Anywhere®
- Sybase SQL Server

Ještě než začneme používat Database Toolbox společně s některou z výše uvedených databází, je nutné nastavit datové zdroje. Tyto zdroje dat pak nastavují tzv. ODBC nebo JDBC ovladače, které slouží jako mezičlánek mezi klientskou aplikací a databází.

3.2.2 ODBC a JDBC ovladače

Prostřednictvím Ovladačů ODBC (Open Database Connectivity) a JDBC (Java Database Connectivity) nastavujeme zdroje dat. Tyto zdroje dat se skládají ze dvou částí, a to z dat, ke kterým přistupuje Database Toolbox, a informací, potřebných k nalezení těchto dat, tzn. ovladačů, adresářů, serverů. Zdroje dat se zpravidla instalují během instalace databázového systému, ale někdy jej musíme nainstalovat ručně.

Na platformě Windows lze využít jak ovladače ODBC, tak JDBC, zatímco platforma UNIX podporuje pouze ovladače typu JDBC. Database Toolbox je založen na Javě a využívá JDBC/ODBC bridge, který spojuje ODBC driver s databází, která je automaticky instalovaná jako součást MATLAB JVM. Následující obrázek 13 ukazuje spolupráci ovladačů s Database Toolboxem.



Obrázek 13 - Popis komunikace Database Toolboxu s databází pomocí ovladačů [12]

3.2.3 Nastavení zdroje dat s použitím ODBC

Jako ukázka je použito nastavení zdroje dat pomocí ODBC ovladače na platformě Windows. Existují opět dvě možnosti, jak provést nastavení zdroje dat. Lze je nastavit přímo ve Windows, kde se v nástrojích pro správu vyskytuje Správce zdrojů dat ODBC, nebo přímo v MATLABU se pomocí nástroje VQB dá vyvolat totožná nabídka jako v prvním případě. Následující obrázek 14 ukazuje dialogové okno pro nastavení zdroje dat.

Obrázek 14 - Správce zdrojů ODBC ve Windows

Po zobrazení výše zmíněného dialogu je možno na kartě Uživatelské DSN[8] přidat nový datový zdroj ze seznamu nainstalovaných ODBC ovladačů. Po přidání daného ovladače se zobrazí dialogové okno s nastavením, které spočívá ve vyplnění údajů použitého databázového systému. Zadá se jméno datového zdroje, přihlašovací údaje a název schématu, s kterým chci pracovat. Toto okno se liší použitím různých ODBC ovladačů. Obrázek 15 ukazuje editační dialog, který používá ODBC ovladač MySQL ODBC 3.51 Driver, který je použit pro spojení s databází MySQL. Tento

[8] Data Source Name

ovladač nebyl v seznamu ODBC zdrojů, proto bylo nutné tento ovladač vyhledat a nainstalovat.

Obrázek 15 - Dialog pro nastavení zdrojů dat pomocí MySQL ODBC connector 3.51

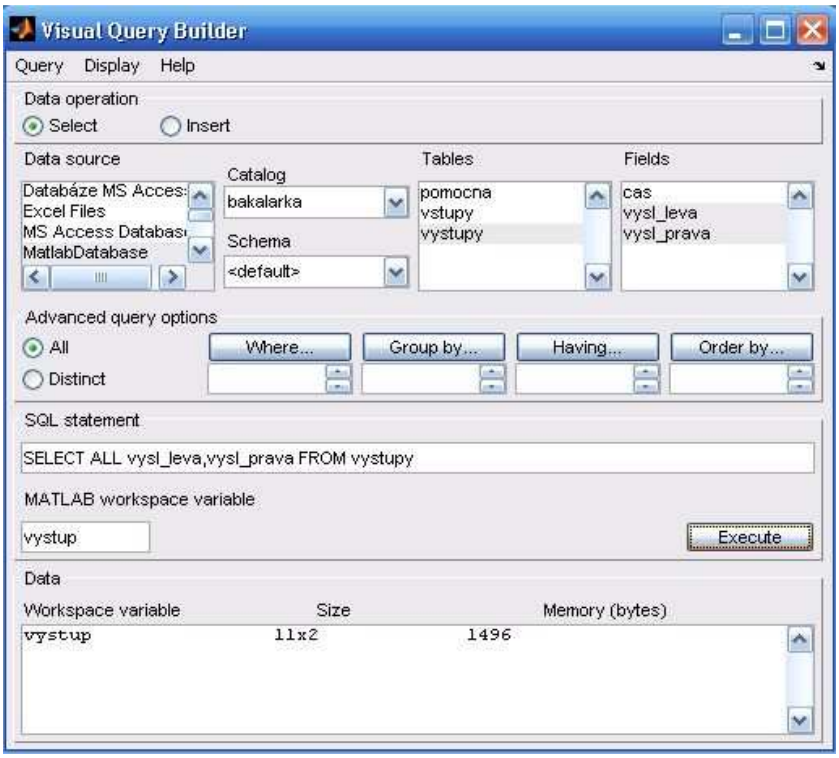
Po nastavení tohoto datového zdroje již nic nebrání pohodlné práci s databázemi v MATLABu, prostřednictvím Database Toolboxu a databázového systému MySQL.

3.2.4 Spolupráce MATLABu s databází

Jak již bylo uvedeno výše, Database Toolbox spolupracuje s databázemi buď prostřednictvím svých funkcí, nebo pomocí nástroje Visual Query Builder, který můžeme spustit příkazem querybuilder v příkazovém řádku MATLABu.

Visual Query Builder

Visual Query Builder, dále jen VQB, je jednoduché uživatelské rozhraní (GUI), které umožňuje snadnou práci s databázemi v MATLABu. Jednoduché proto, že pro používání tohoto nástroje není nutné znát syntaxi databázových dotazů a postačí pouze základní znalosti v této oblasti. Při práci s VQB je totiž možno vybrat si z nabídky datový zdroj, zvolit schéma, tabulku a sloupec. Automaticky se vytvoří požadovaný dotaz, aniž by byla nutná znalost jazyka SQL. Pro představu, jak vypadá prostředí VQB, je uveden obrázek 16, kde je zachyceno hlavní okno.



Obrázek 16 – Pracovní prostředí VQB

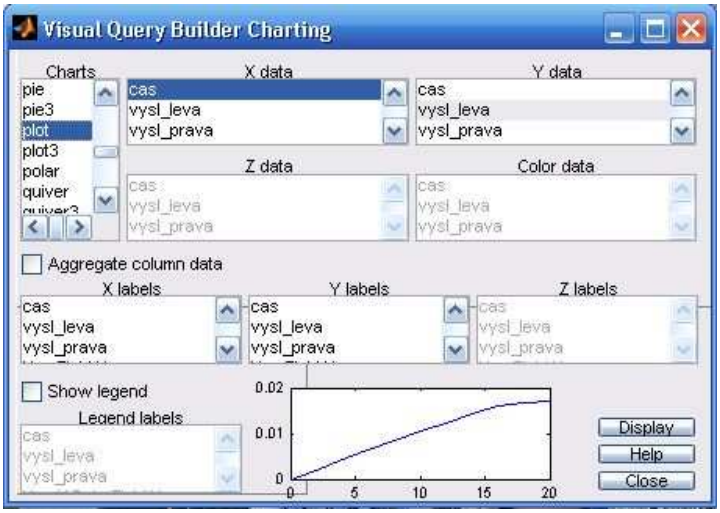
Na obrázku 16 je viditelný postup, jak například vybrat data z databáze. Nejprve došlo k výběru operace SELECT pro výběr dat, následně byl zvolen zdroj dat, který jsem v předchozí kapitole nastavil, a vyberu schéma, tabulku a sloupec. V tomto příkladu dochází k získání dat ze schématu *bakalarka*, tabulky *vystupy* a ze sloupců *vysl_leva* a *vysl_prava.* Po výběru těchto dvou sloupců se automaticky vytvoří dotaz. Nakonec už jen zbývá definovat proměnnou, do které se výsledky dotazu zapíší, a může nastat jeho spuštění. Když se dotaz úspěšně provede, zapíší se do proměnné potřebná data, se kterými můžeme v MATLABu pracovat. Pro náročnější dotazy je možné použít klauzule WHERE, GROUP BY, HAVING nebo ORDER BY, kde se opět pouhým klikáním nastavují příslušné podmínky, které by měl dotaz splňovat.

Kdyby bylo potřeba data do databáze uložit, postup je podobný jako v případě výběru dat pomocí SELECTu. Stačí pouze označit INSERT pro vložení dat, definovat proměnnou, z které se mají data uložit, a vybrat sloupce tabulky, které mají být během dotazu ovlivněny. Stejně jako v případě SELECTu VQB automaticky vygeneruje dotaz a naším úkolem je tento dotaz pouze spustit.

Jsou tedy načtená data z databáze, ale je potřeba je zobrazit nebo vykreslit graf. VQB vychází uživatelům vstříc a obsahuje několik nástrojů pro zobrazení dat jak v grafické, tak i v textové formě ihned po jejich načtení. Tyto nástroje se nachází v nabídce *Display.*

Formy zobrazení dat pomocí VQB

- **Data –** zvolením této volby v nabídce *Display* dojde k zobrazení dat v jednoduchých sloupcích s možností další úpravy nebo uložení do libovolného formátu
- **Chart –** tato volba je velice zajímavá, protože nabízí z načtených dat přímo vykreslit různé typy grafů, nastavit hodnoty os a uložit ho do různých formátů. Ukázku tohoto nástroje zachycuje obrázek 17**.**



Obrázek 17 - Nástroj Chart pro vykreslení grafu z načtených dat

- **Report –** výběrem této volby je možno vytvářet přehledné reporty ve formátu HTML a zobrazit data v přehledné formě na webu např. prostřednictvím tabulek. Tyto reporty je však nutné nejprve vytvořit v *Report Generátoru*, který je k dispozici také v nabídce *Display*

Základní nastavení VQB

Nástroj VQB obsahuje potřebná nastavení, která jsou nutná pro správnou reprezentaci načtených dat apod. Konkrétně umožňuje nastavit reprezentaci hodnot NULL, načtených z databáze v MATLABu, formát načtených dat nebo chybová hlášení. Pro nejdůležitější nastavení, kterým je formát načtených dat, nabízí VQB výběr

ze tří voleb, tzn., že data mohou být reprezentována jako buněčná pole, číselný formát nebo formou struktury.

Výhody VQB

- Jednoduché a příjemné uživatelské rozhraní
- Pohodlná práce při sestavování SQL dotazů
- Umožňuje vytvořit i složité dotazy během několika sekund a pár kliknutí
- Možnost zobrazení výsledků dotazů formou reportů nebo grafů
- Generování MATLAB m-file, který obsahuje funkce Database Toolboxu

Nevýhody VQB

- Zabudované jsou pouze dotazy pro výběr a vkládání dat SELECT a INSERT
- Nelze tedy použít pro mazání nebo aktualizaci dat pomocí příkazů UPDATE nebo DELETE
- Nemožnost exportovat binární data
- Pokaždé, když je potřeba pracovat s databází, musí být spuštěné GUI
- Ne vždy je použitelný, např. při práci s databází v SIMULINKu lze použít pouze funkce Database Toolboxu a napsat si vlastní m-file.

Práce s databází prostřednictvím funkcí Database Toolboxu

Jak vyplývá z předchozího textu, nástroj VQB je sice jednoduchý a snadno se s ním pracuje, ale přece jenom častěji se využívá skript, který bude muset být vytvořen pomocí vestavěných funkcí Database Toolboxu.

Skripty, ve kterých se definují funkce pro práci s databází, se nazývají M-file a jsou to standardní skripty MATLABu. Při práci s funkcemi Database Toolboxu je nutná alespoň jejich částečná znalost, která je nezbytná k jejich správnému používání. Používání těchto funkcí není tak omezené, jak tomu bylo ve VQB, ale rozšířené a vhodné pro pokročilou práci s databází. Lze pomocí nich například načíst velké objemy dat, dynamicky je importovat nebo je používat pro práci s binárními daty nebo pro přístup k metadatům. V následujícím odstavci budou popsány některé základní funkce pro práci s databází.

Popis základních funkcí Database Toolboxu

- **Database –** jedná se o funkci, která slouží k navázání spojení s existující databází, kde prvním parametrem je název datového zdroje, druhým a třetím parametrem jsou přihlašovací údaje do databázového systému.
    - ***Použití:*** *spojeni=database('NazevZdrojeDat','UzivJmeno','Heslo')*
- **Close –** tato funkce plní dva účely, kterými jsou uzavření aktuálně vytvořeného spojení a uzavření otevřených kurzorů. Kurzory slouží k načítání údajů z databáze do proměnných, přičemž práce s kurzory je obdobná jako se soubory.
    - ***Použití:*** *close(spojeni)* nebo *close(kurzor)*
- **Setdbprefs –** je to funkce, která slouží k nastavení preferencí např. formátu dat nebo chybových hlášení při exportu z databáze do MATLABu. Má dva parametry, první určuje, co nastavit, a druhý, jak to mám nastavit.
    - ***Použití:*** *setdbprefs ('DataReturnForma','numeric')*
- **Insert/fastinsert** – obě tyto funkce slouží k vkládání dat z MATLABu do databáze. Funkce fastinsert je novější, výkonnější a podporuje více datových typů než klasický insert. Syntaxe obou funkcí je totožná a to tak, že prvním parametrem je spojení, druhým je tabulka, třetí parametr značí sloupec a poslední parametr určuje data.
    - ***Použití:*** *fastinsert(spojeni, 'Tabulka', NázvySloupců, Data)*
- **Update –** touto funkcí je možno pohodlně aktualizovat záznamy v tabulkách. Má pět parametrů, kde první je spojení, druhý název tabulky, třetí obsahuje názvy sloupců, čtvrtý data a konečně poslední podmínky, jaká data budou změněna.
    - ***Použití:*** *update (spojeni, 'Tabulka', NázvySloupců, Data, 'Podmínky')*
- **Exec** – je funkce, která provede dotaz a výsledek uloží do kurzoru. Parametry této funkce jsou spojení a příslušný SQL dotaz, který chcete použít.
    - ***Použití:*** *kurzor = exec(spojeni, 'DotazSQL')*
- **Fetch -** tato funkce má dvě varianty a to cursor.fetch a database.fetch. První z nich slouží k načtení dat z kurzoru vytvořeného pomocí funkce exec a druhý je podobný jako funkce exec, ale s tím rozdílem, že zároveň vykoná příkaz a načte data.

- ***Použití cursor.fetch:*** *kurzor = fetch(kurzor)*
- ***Použití database.fetch:*** *vysledek = fetch(spojeni, 'DotazSQL')*

Zde bylo uvedeno několik základních funkcí, jejichž znalost umožňuje vytvářet jednoduché, ale i složitější skripty pro práci s databází. Samozřejmě výčet těchto funkcí není konečný, ve skutečnosti jich je mnoho a postupný popis všech funkcí by byl zbytečný. Následující obrázek 18 zachycuje ukázku skriptu pro získání dat z databáze.

```
1     %vytvoreni spojeni
2 -   spojeni = database('MatlabDatabase','root','vertrigo');
3     %provedeni a ulozeni dat do kurzoru
4 -   kurzor = exec(spojeni,'SELECT * FROM pomocna');
5     %nastaveni formatu zobrazenych dat
6 -   setdbprefs('DataReturnFormat','numeric');
7     %nacteni dat z kurzoru
8 -   kurzor = fetch(kurzor);
9     %zobrazeni dat
10 -   a = kurzor.Data;
11    %zavreni kurzoru
12 -  close(kurzor);
13    %zavreni spojeni
14 -  close(spojeni);
15
```

Obrázek 18 - Skript pro získání dat z databáze

Výhody použití funkcí Database Toolboxu

- Možnost aktualizace a mazání dat formou dotazu
- Lze pomocí nich vytvořit skripty a funkce a opakovaně je používat
- Načtení velkého objemu dat pomocí funkce fetch
- Práce s binárními daty nebo metadaty

Nevýhody použití funkcí Database Toolboxu

- Nutná znalost funkcí Database Toolboxu a práce s nimi
- Pro běžné uživatele mnohdy nepřehledné a složité

3.3 Real Time Toolbox

Jedním z předpokladů při propojení výstupů simulace s databází je, aby simulace probíhala v reálném čase. Proto je vhodné, zmínit se o Real Time Toolboxu trochu podrobněji.

Real Time Toolbox je jedním z modulů MATLABu a slouží pro spojení SIMULINKu s reálným světem. Umožňuje přístup k vnějším analogovým a digitálním

signálům. Knihovnu bloků je možno využít pro experimenty se zpracováním signálů, seřízením regulátorů a s dalšími podobnými úkoly v SIMULINKu [13]. Mimo to ho lze využít k zajištění k výpočtu v SIMULINKu v reálném čase. Při práci s Real Time Toolboxem nejsou potřeba téměř žádné znalosti, které se týkají programování použitého hardware, stačí pouze vložit příslušný blok do simulace a spustit simulaci.

V současné době je na trhu verze Real Time Toolbox 4.01 a je možné ji zakoupit u firmy HUMUSOFT. Tento toolbox je využit při realizaci spojení simulace s databází v reálném čase.

4 Praktická část

V teoretické části byly vysvětleny základní pojmy databázových systémů, proběhlo seznámení s prostředím MATLABu a především byl představen Database Toolbox, společně s jeho možnostmi spolupráce s databázovým systém. Proto je vhodná doba využít tyto teoretické poznatky v praxi.

Hlavním cílem praktické části je tedy navržení vhodných m-funkcí, které jsou nutné k načtení nebo uložení dat do databáze, navázání spojení MATLABu s databází MySQL a propojení těchto m-funkcí s nelineárním matematickým modelem hydraulicko-pneumatické soustavy, jehož se využívá při simulacích v programu SIMULINK. Uživatel může simulaci ovládat pomocí webové aplikace, která byla rovněž vytvořena v rámci této práce.

Praktická část je rozdělena do tří kapitol, kde první kapitola se věnuje řešení daného problému v programu MATLAB a SIMULINK. Druhá část se zabývá návrhem databáze a jejím propojením s MATLABem. Poslední, třetí část popisuje implementaci webového uživatelského rozhraní.

4.1 Popis praktického řešení v MATLABu

Tato kapitola objasňuje tvorbu m-funkcí pro komunikaci s databází. Je zde uvedena charakteristika matematického modelu a napojení vytvořených m-funkcí na matematický model v SIMULINKu. Závěrem je popsáno vytvoření skriptu, umožňujícího spouštění simulace na dálku.

4.1.1 Tvorba m-funkcí pro komunikaci s databází

Pro tvorbu m-funkcí je možné použít editor pro tvorbu skriptů a funkcí, který je k dispozici v prostředí MATLABu nebo kterýkoliv jiný textový editor. Pro potřeby této práce byly vytvořeny dvě základní m-funkce, kde první slouží k načtení aktuálních hodnot vstupů z databáze a druhá ukládá data z výstupů matematického modelu do databáze.

Funkce pro získání dat z databáze *GetData.m* je vytvořena univerzálním způsobem, tzn., že se může použít na libovolné tabulky a sloupce. Má dva parametry, kde první udává název sloupce a druhý tabulku, z které se čerpají vstupní hodnoty.

Funkce na začátku ověří, zdali jsou zadány oba parametry, pokud ne, tak funkce vypíše chybovou hlášku a ukončí se. Poté dojde ke spojení pomocí funkce *database*, kde je zadán název zdroje dat a přihlašovací údaje do databáze. Následně se provede funkce *isconnection*, která vrací hodnotu true, pokud se podaří realizovat spojení, jinak vrací false. Po provedení těchto základních úkonů je definován dotaz, jenž vybere hodnoty, kde čas vložení je maximální. Maximální čas je vybrán, protože simulace běží v reálném čase a na vstupy modelu je třeba vkládat aktuální data. Proto je vytvořen vnořený dotaz pomocí dvou operací SELECT a agregační funkce MAX. Funkcí *exec* je dotaz proveden, uložen do kurzoru a po nastavení číselného formátu dat funkcí *setdbprefs* jsou funkcí *fetch* načtena všechna data do proměnné kurzoru. Ta jsou poté vložena pomocí metody *data* do výstupní proměnné. Nakonec je uzavřen kurzor a spojení s databází pomocí funkce *close*.

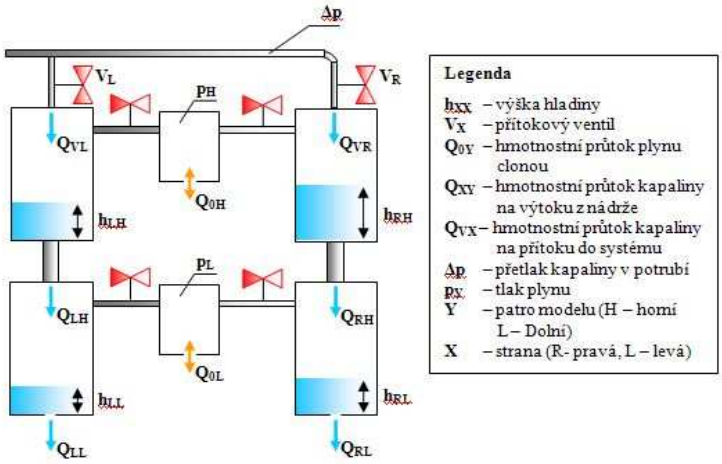
Pro ukládání dat do databáze je napsána m-funkce *Zapis.m*, ale na rozdíl od předchozí funkce je podstatně jednodušší. Tato funkce je také vytvořena univerzálně, proto ji lze použít při práci s různými tabulkami a sloupci. Má pět parametrů, kde první udává proměnnou, z které se data ukládají do databáze. Druhým parametrem je tabulka, s níž se pracuje, a následující tři parametry jsou sloupce, do kterých jsou ukládány jednotlivé hodnoty ze simulačního modelu. Na začátku funkce je opět ověření, zda uživatel zadal správný počet vstupů a pokud tomu tak není, zobrazí se chybová hláška. Potom následuje spojení s databází pomocí funkce *database* s údaji, které jsou popsány výše. Po testu na spojení jsou názvy sloupců, stejně jako jejich hodnoty, uloženy do dvou proměnných jako pole z důvodu lepšího použití ve funkci pro zápis dat do databáze. Posléze je použita funkce *fastinsert*, která kromě parametru spojení obsahuje také použitou tabulku a pomocné proměnné. Na závěr je pak stejně jako u předchozí funkce ukončeno spojení s databází.

4.1.2 Realizace modelu v SIMULINKu

Jako prostředek pro testování spojení MATLABu s databázemi je použit matematický model hydraulicko-pneumatické soustavy, který je vytvořený v SIMULINKu.

Charakteristika použitého matematického modelu

Matematický model, který je použit jako prostředek pro ověření spolupráce MATLABu s databází, je tvořen čtyřmi hydraulickými a dvěma pneumatickými nádržemi. Hydraulické nádrže tvoří dvě patra po dvou nádržích, kde obě nádrže v každém patře jsou propojeny jednou pneumatickou nádrží. Všechny hydraulické nádrže jsou na dně opatřeny odtokovým otvorem. Pneumatické nádrže jsou opatřeny malým otvorem, pomocí něhož dochází k vyrovnání tlaku plynu na jednotlivých patrech. Tlak plynu v jednotlivých patrech je totiž závislý na změnách výšky hladiny. Přívod vody do těchto nádrží je realizován jedním potrubím a pomocí ventilů regulován průtok do jednotlivých nádrží. V závislosti na otevření ventilů a výšce hladin kapaliny v horních nádržích se mění i hmotový průtok kapaliny na odtoku. Odtok horních nádrží tvoří přítok kapaliny do dolních nádrží. Obrázek 19 zachycuje propojení hydraulicko-pneumatické soustavy. Podrobný popis modelu je k dispozici v diplomové práci [14].



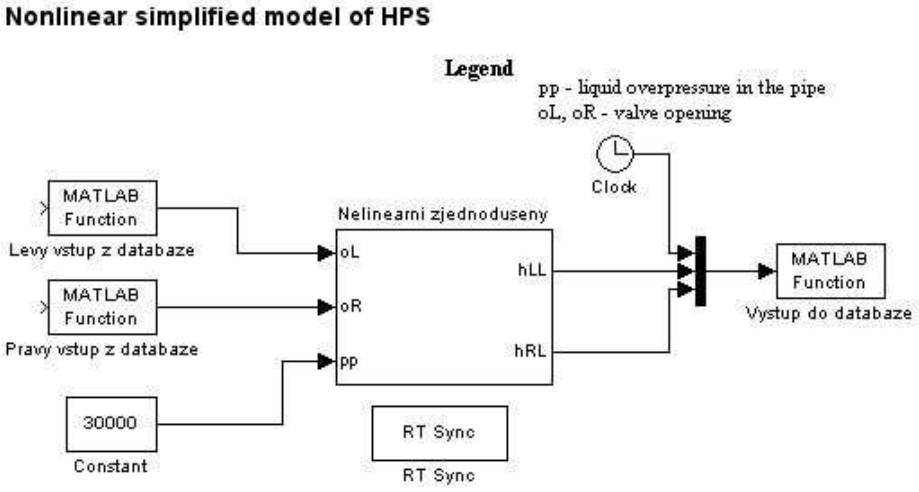
Obrázek 19 - Hydraulicko-pneumatická soustava se čtyřmi nádržemi

K simulacím je využito zjednodušeného matematického modelu této soustavy. Vstupy do simulace představují otevření levého a pravého ventilu. Třetí vstup (tlak kapaliny v přívodním potrubí), kterým soustava disponuje v této práci, využit není. Výstupy soustavy jsou výšky hladiny kapaliny v levé a pravé dolní nádrži. Prostřednictvím webového rozhraní se nastavuje otevření ventilů. Uživatelem nastavované vstupy se přes webové rozhraní ukládají do databáze, odkud jsou předány do simulace. Výstupy simulace jsou v pravidelných intervalech ukládány do databáze, odkud jsou přístupné dalším aplikacím, tedy i webovému rozhraní. Aby se simulace

chovala jako reálné zařízení, a tedy bylo možno ji ovládat přes webové rozhraní, bylo nutné při simulacích využívat Real Time Toolboxu.

Napojení m-funkcí na simulační model v SIMULINKu

Po stručné charakteristice použitého modelu je možné přistoupit k jeho spojení s vytvořenými m-funkcemi. Docílí se toho použitím speciálního bloku s názvem *MATLAB function*, který je součástí standardní nabídky bloků a je možno najít ho v oddělení *User-Defined-Functions*. Tento blok je využit celkem třikrát. První dva slouží jako vstupní bloky, kde je definována m-funkce pro načtení dat z databáze. Zbývající blok tvoří výstup do databáze Je tvořen vektorem tří hodnot, kde první je čas, který odpovídá simulačnímu času, a zbylé dvě hodnoty udávají výšky levé a pravé nádrže. Na obrázku 21 je schéma, které názorně ukazuje napojení nelineárního modelu na databázi.



Obrázek 20 - Nelineární model HPS napojený na databázi

Je třeba zajistit, aby simulace probíhala v reálném čase. Toho lze docílit například pomocí Real Time Toolboxu, který vyvinula firma HUMUSOFT. V této práci je použita verze 4.0.

4.1.3 Návrh skriptu pro vzdálené spuštění simulace

K zajištění možnosti spouštění simulace z webového rozhraní je nutné vytvořit v prostředí MATLABu skript, který s využitím funkcí Database Toolboxu zajistí na straně MATLABu tuto funkci.

Skript tvoří tzv. nekonečnou smyčku. Znamená to, že běží neustále a reaguje na požadavek spouštění simulace provedený uživatelem z webového rozhraní.

Skript se nazývá *smycka.m* a je vytvořen v editoru, který je dostupný v MATLABu. Tento skript využívá pro svou činnost tabulku *pomocna*, kde je uchován stav, v jakém se simulace nachází, a délka simulace. Stav simulace je 1, pokud má být simulace spuštěna, nebo 0 v případě, že simulace nemá být spuštěna. Nejdříve tento skript naváže spojení s databází pomocí funkce *database* a následně funkcí *isconnection* zkontroluje, zda spojení bylo úspěšně navázáno. Pokud vše proběhlo bez problémů, načte se soubor *parametry.m*, který obsahuje všechny potřebné parametry matematického modelu. Pak již běží nekonečný cyklus, kde v každém cyklu je načten stav simulace z tabulky *pomocna* prostřednictvím funkce *GetPomocna*. Provede se vyhodnocení, a pokud je stav roven jedné, dojde k načtení délky simulace a příkazem *sim* k jejímu spuštění. Po dokončení spuštěné simulace jsou funkcí *ZapisPomocna* uloženy nulové hodnoty do tabulky *pomocna*. Tyto nulové hodnoty způsobí, že smyčka bude běžet dál s tím, že je možné spouštět další simulační experimenty. Při běhu nekonečné smyčky docházelo k velkému vytěžování procesoru, proto bylo nutné smyčku na jednu sekundu pozastavit pomocí funkce *pause*.

4.2 Databáze

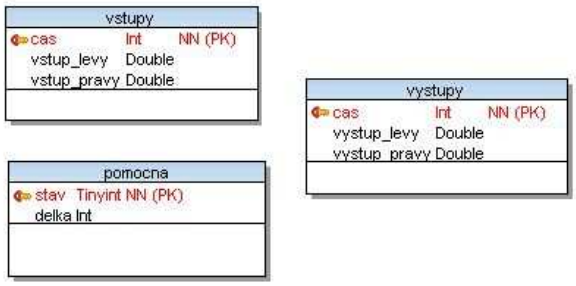
Tato kapitola se zabývá návrhem struktury databázových tabulek a realizací spojení mezi MATLABem a databází MySQL. Jak už bylo řečeno, použitým databázovým systémem je MySQL databáze, která je dnes s oblibou využívána v různých odvětvích. Tato databáze je součástí balíku VertrigoServ, který je volně k dispozici.

4.2.1 Návrh databázových tabulek

Tabulky jsou vytvořeny v uživatelském rozhraní PhpMyAdmin, které je stejně jako MySQL databáze součástí balíku VertrigoServ a slouží k pohodlné správě tabulek.

Pro potřebu ukládání a načítání hodnot během simulace jsou vytvořeny celkem tři tabulky. První tabulka se jmenuje *vstupy a* skládá ze tří sloupců. Těmito sloupci jsou čas zadání vstupní hodnoty, jenž tvoří primární klíč. Další sloupce obsahují hodnoty levého a pravého vstupu. Čas zadání se počítá od počátku simulace. Z této tabulky jsou načítány vstupy do simulačního modelu. Druhá tabulka s názvem

vystupy je stejně jako tabulka vstupů tvořena také třemi sloupci. První sloupec obsahuje čas simulace, pro který se ukládají hodnoty do databáze, a představuje primární klíč tabulky. Zbylé dva sloupce uchovávají hodnotu dolní levé a pravé výšky hladiny. Hodnoty výstupů simulace se do této tabulky ukládají každé dvě sekundy. Poslední tabulka se nazývá *pomocna* a je tvořena dvěma sloupci a jedním řádkem, kde první sloupec může nabývat hodnot 0 nebo 1 a využívá se jej pro spouštění simulace na dálku, kde hodnota 1 představuje požadavek na spuštění simulace. Druhý sloupec obsahuje délku simulace. Následující obrázek 20 ukazuje schéma tabulek vytvořených v Toad Data Modeleru 3.



Obrázek 21 - Schéma tabulek vytvořené v Toad Data Modeleru 3

4.2.2 Spojení MATLABu s databází MySQL

Jak bylo napsáno v teoretické části, spojení s databází je realizováno prostřednictvím ODBC/JDBC ovladače. Nástroj VertrigoServ implicitně neobsahuje tento ovladač pro databázi MySQL, proto je třeba vyhledat tento ovladač a nainstalovat ho. Byl vyhledán ODBC ovladač mysql-connector-odbc-3.51.27, který je volně ke stažení s licencí GPL. Po úspěšné instalaci tohoto ovladače je prostřednictvím nástroje pro přidání datových zdrojů přidán nový datový zdroj pro databázi MySQL, který je součástí operačního systému Windows. Tento postup, včetně nastavení, je uveden v teoretické části, proto není nutné se jím zabývat znovu.

Po úspěšném vložení a správném nastavení datového zdroje je již databáze MySQL schopná komunikovat s MATLABem prostřednictvím Database Toolboxu a používat jeho funkce pro navázání spojení a ukládání dat do vytvořených tabulek.

4.3 Webové uživatelské rozhraní

Webové rozhraní je jako grafické rozhraní, umožňující uživateli sledovat a ovládat řízený model na dálku. Toto rozhraní je stejně jako MATLAB propojeno s databází MySQL. Tím se vytvoří tzv. virtuální spoj mezi simulací a webovým rozhraním, pomocí kterého je možné ovládat simulace na dálku.

Při realizaci webového rozhraní byl použit značkovací jazyk HTML v kombinaci se skriptovacím jazykem PHP. Pro vytvoření grafických stylů je využívána technologie CSS. Své uplatnění, i když jen malé, má i JavaScript, který slouží pro automatické obnovení grafů. Jako webový server je použit Apache ver. 2.0, který je společně s jazykem PHP ver. 5 součástí balíku VertrigoServ.

4.3.1 Popis souborů

Hlavní kostru aplikace tvoří soubor *index.php*, kde je vytvořena podoba základního layoutu s rozmístěním jednotlivých elementů, včetně menu, záhlaví nebo zápatí. Dalším souborem je *funkce.php*, kde je implementována funkce pro připojení k databázi MySQL. Asi nejdůležitějším souborem je *nastaveni.php*, který slouží pro ovládání simulace. V tomto souboru jsou implementovány formuláře pro zadávání dat, ověření správnosti zadaných dat, přepočty reálného času na simulační a především spolupráce s databází při spouštění simulace nebo práci s daty. Výpis výstupních hodnot zajišťuje soubor *vysledky.php*, který čerpá data vložená do databáze a zobrazuje je v přehledné formě uživateli. Grafické znázornění výsledků má na starosti soubor *grafy.php*, kde jsou vytvořeny dva obrázky ze souborů *graf_levy.php* a *graf_pravy.php*. V těchto souborech jsou implementovány grafy pro levý, respektive pravý výstup jako obrázek PNG. Tvorbu obrázků umožňuje grafická knihovna GD, která je součástí skriptovacího jazyka PHP. Pomocí funkcí této knihovny lze vytvořit graf a následně z něj udělat obrázek. Soubor *grafy.php* také obsahuje JavaScript pro automatickou obnovu grafů bez nutnosti měnit obsah stránky. Automatická obnova probíhá každou sekundu po dobu sto sekund, pak se obnova ukončí. Všechna stylová nastavení jsou definována v souboru *styly.css*.

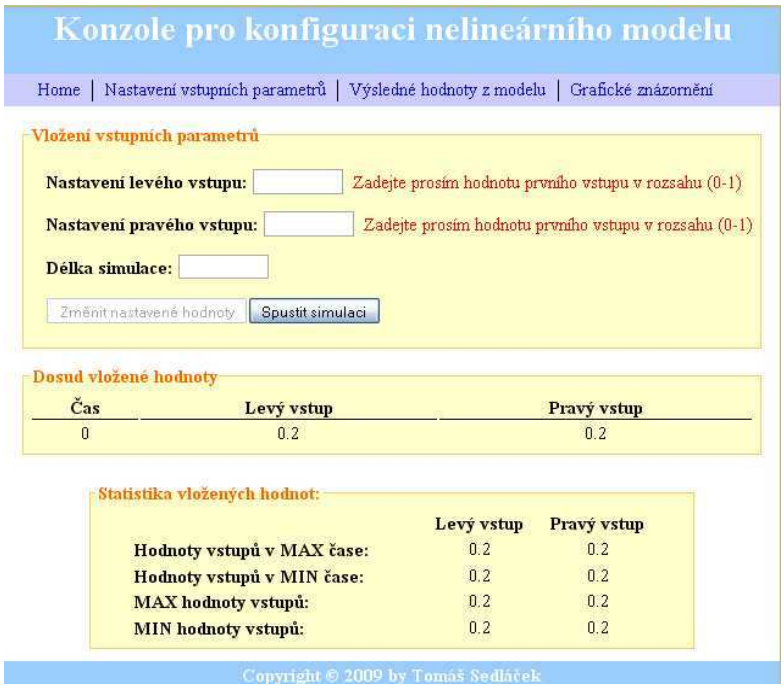
4.3.2 Funkčnost webové aplikace

Správnou funkčnost webové aplikace je nutné zajistit spuštěním programu MATLAB a skriptu *smycka.m*. Zároveň je nutné zkontrolovat, zda v tabulce *pomocna* jsou vloženy nulové hodnoty.

Spuštěním webové aplikace v nějakém prohlížeči se zobrazí úvodní stránka, která uživatele uvítá a v krátkosti seznámí s použitelností aplikace. Tuto stránku znázorňuje obrázek 22.

Obrázek 22 - Úvodní obrazovka webové aplikace

Po seznámení s úvodní obrazovkou je možno přejít k samotnému nastavení vstupů a spuštění simulace. Poklepáním na položku menu *Nastavení vstupních parametrů* dojde k načtení stránky pro ovládání simulace. Pokud již běží v MATLABu skript s nekonečnou smyčkou, nic nebrání tomu nastavit levý (pravý) vstup, délku simulace a následně simulaci spustit. Vstupy se musejí zadat v rozmezí 0-1, kde 0 znamená, že ventil je uzavřený, a 1 představuje maximální otevření ventilu. Aplikace je na toto zadávání připravena a nedovolí uživateli vložit jinou hodnotu než v tomto rozmezí. Délka simulace se určuje jako celé číslo v sekundách. Po zmáčknutí tlačítka *Spustit simulaci* se vymažou data z tabulek *vystupy* a *vstupy* a zapíše se hodnota 1, společně s délkou simulace, do tabulky *pomocna*, což způsobí, že spuštěný skript v MATLABu přijme příznak 1 a spustí simulaci. Zároveň dojde ke zneaktivnění tohoto tlačítka a aktivuje se tlačítko *Změnit vstupní parametry*. Nyní za běhu simulace je možné měnit vstupní hodnoty bez možnosti měnit délku simulace, která je stejně jako tlačítko na spouštění simulace neaktivní. Po uplynutí času simulace je tlačítko pro její spouštění automaticky aktivováno a vše je připraveno pro další spouštění simulace. To umožňuje uživateli opakovaně provádět simulační experimenty bez nutnosti být přímo v kontaktu se simulačním modelem v MATLABu. Pro ilustraci je uveden obrázek 23, který ukazuje obrazovku pro nastavení vstupních parametrů.



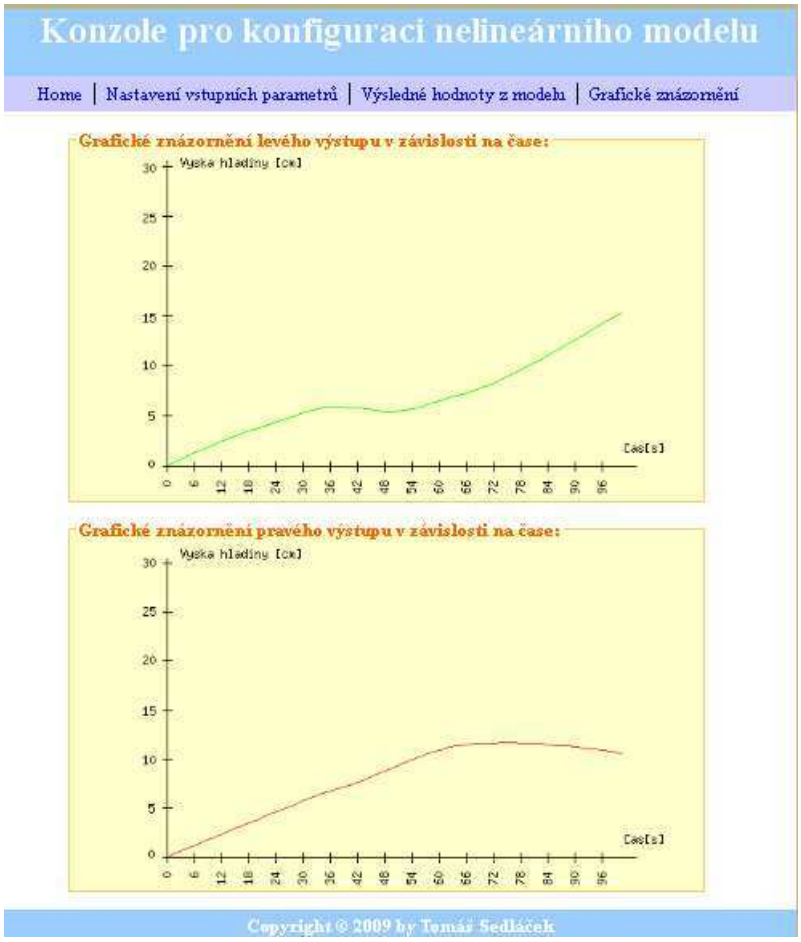
Obrázek 23 – Obrazovka pro nastavení a spuštění simulace

Jednou ze základních funkcí, kterou webové rozhraní nabízí, je zobrazení výstupů simulace. K tomu slouží položka menu *Výstupní hodnoty z modelu*. Na této obrazovce jsou zobrazeny v přehledné formě výsledky z levého a pravého výstupu, přesněji výšky levé a pravé nádrže matematického modelu. Společně s nimi je zde zachycen i simulační čas. Je zde také možnost vyprázdnit tabulku ručně příslušným tlačítkem na konci stránky. Pro lepší představu je uveden obrázek 24.

Obrázek 24 - Obrazovka zobrazující výsledné hodnoty z modelu

Zobrazení výstupů ze simulačního modelu v textové podobě nemusí být pro uživatele dostatečně čitelné. Proto jako poslední funkcí webové aplikace je zobrazení výstupních hodnot v grafické formě. Stačí, když se v menu vybere položka *Grafické*

znázornění a načte se stránka, kde jsou zobrazeny dva grafy, první pro levý a druhý pro pravý výstup. Jak již bylo zmíněno výše, tyto grafy jsou zde vloženy jako obrázky, které byly vytvořeny pomocí grafické knihovny GD. Tyto grafy udávají závislost výšky hladiny obou nádrží, které se mění v čase. Při spuštění simulace se grafy pomocí JavaScriptu automaticky každou sekundu obnovují po dobu sto sekund. Jako malá ukázka těchto grafů je uveden obrázek 25.



Obrázek 25 - Grafické znázornění výstupních hodnot

5 Závěr

Cílů, vytyčených v úvodu práce, bylo úspěšně dosaženo. Podařilo se vytvořit skripty umožňující propojení a výměnu dat mezi MATLABem a databází MySQL v reálném čase. Za účelem ovládání simulace na dálku bylo vytvořeno webové rozhraní, které je schopné opakovaně spouštět simulaci a během jejího konání umožňuje v reálném čase měnit vstupní parametry. Webová aplikace uživateli rovněž zobrazuje průběhy výstupů modelu. Ty si může uživatel zobrazit v grafické podobě pomocí grafů, nebo textové podobě ve formě tabulek.

Tato práce ukázala, že by v budoucnu bylo možno využít MATLABu jako nástroje, který by umožnil vytvořit pro studenty virtuální laboratoř. Ta by umožnila nahradit reálné laboratorní modely, modely matematickými, čímž by došlo k úspoře místa potřebného k výuce. Jak bylo naznačeno v teoretické části, nabízí propojení MATLABu s databázemi možnost využít výpočetní síly MATLABu pro analýzu a zpracování dat naměřených na reálných zařízeních. Tohoto by se dalo rovněž využít při výuce. Stačí v budoucnu zajistit propojení reálných zařízení s některým z databázových systémů.

Použité zdroje

[1] *Wikipedia : Databáze* [online]. 2004, 22.2.2009 [cit. 2009-03-05]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Datab%C3%A1ze>.

[2] TELNAROVÁ , Zdeňka, LUKASOVÁ, Alena, MATULA, Petr. *Úvod do databází*. 1. vyd. Ostravská univerzita v Ostravě : [s.n.], 1999. 123 s. ISBN 80-7042-784-1.

[3] *Wikipedia : Systém řízení báze dat* [online]. 2005, 27.2.2009 [cit. 2009-03-05]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Syst%C3%A9m_%C5%99%C3%ADzen%C3%AD_b%C3%A1ze_dat>

[4] *Wikipedia : Relační model* [online]. 2004 , 25.1.2009 [cit. 2009-03-10]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Rela%C4%8Dn%C3%AD_model>

[5] *Manualy.net : Teorie relačních databází: Integritní omezení* [online]. 2006 , 10.9.2006 [cit. 2009-03-11]. Dostupný z WWW: <http://www.manualy.net/article.php?articleID=15>

[6] *Manualy.net : Teorie relačních databází: Normalizace* [online]. 2007 , 2.8. 2007 [cit. 2009-03-11]. Dostupný z WWW: <http://www.manualy.net/article.php?articleID=13>

[7] POKORNÝ, Jaroslav, HALAŠKA, Ivan. *Databázové systémy*. 2. přeprac. vyd. ČVUT Praha : [s.n.], 2004. 148 s. ISBN 80-01-02789-9

[8] *Wikipedia : Relační databáze* [online]. 2004 , 7.3.2009 [cit. 2009-03-17]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Rela%C4%8Dn%C3%AD_datab%C3%A1ze>

[9] VOBECKÝ, Antonín. *Návrh relačních databázových systémů*. SPŠE Praha : [s.n.], 2005. 130 s.

[10] DUŠEK, František. *MATLAB a SIMULINK : úvod do používání*. 2. rozš. vyd. Pardubice : [s.n.], 2002. 158 s. ISBN 80-7194-475-0.

[11] *The MathWorks : Database Toolbox: Supported Databases* [online]. c1984-2009 [cit. 2009-03-31]. Dostupný z WWW: <http://www.mathworks.com/access/helpdesk/help/toolbox/database/ug/bq89k8o-1.html#bq89k8o-4>.

[12] *The MathWorks : Database Toolbox: About Data Sources and Database Drivers* [online]. c1984-2009 [cit. 2009-04-01]. Dostupný z WWW: <http://www.mathworks.com/access/helpdesk/help/toolbox/database/gs/braiey2-1.html#bre040n-1>.

[13] *Humusoft : Real Time Toolbox* [online]. c1991-2009 [cit. 2009-04-27]. Dostupný z WWW: <http://www.humusoft.com/produkty/rtt/index.php?lang=cz&p1=1&p2=5>.

[14] ŠKRABÁNEK, Pavel. *Matematický model hydraulicko-pneumatické soustavy*. [s.l.], 2004. 51 s. Diplomová práce.

[15] CASTAGNETTO, Jesus, et al. *Programujeme PHP profesionálně*. 2. aktualiz. vyd. [s.l.] : [s.n.], 2002. 656 s. ISBN 80-7226-310-2.

Přílohy

Všechny přílohy, včetně zdrojových kódů a bakalářské práce v elektronické podobě jsou k dispozici na přiloženém CD.