

Vývoj aplikace SQLCommander - nástroj pro správu databázových systémů

Developing SQLCommander application – tools for managing
database systems

Bc. Jiří Novák

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jiří NOVÁK**
Osobní číslo: **A09738**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Bezpečnostní technologie, systémy a management**

Téma práce: **Vývoj aplikace SQLCommander – nástroj pro správu databázových systémů**

Zásady pro vypracování:

1. Analyzujte problematiku a vypracujte literární rešerši na dané téma.
 2. Navrhněte strukturu aplikace pro správu databázových systémů.
 3. Vytvořte aplikaci, která bude umožňovat zobrazení informací ze systémového katalogu databází (seznam tabulek, jejich sloupce, primární klíče atd.).
 4. Vytvořte další možnosti aplikace – spouštění jednotlivých SQL příkazů, editace databázových struktur, komfortní zadávání SQL příkazů se zvýrazněním syntaxe SQL, spouštění skriptů podle definovaného formátu atd.
 5. Popište způsob implementace systému.
-

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. GROFF, J. M. SQL kompletní průvodce. Brno: Computer Press, 2005. ISBN 80-251-0369-2.
2. MOLINARO, A. SQL – Kuchařka programátora. Brno: Computer Press, 2009. ISBN 978-80-251-2617-2.
3. AGARWAL, V. V. Databáze v C 2008, Průvodce programátora. Brno: Computer Press, 2009. ISBN 978-80-251-2309-6.
4. GUNNERSON, E. Začínáme programovat v C. Brno: Computer Press, 2001. ISBN 80-7226-525-3.
5. ROBINSON, S. C Programujeme profesionálně. Brno: Computer Press, 2003. ISBN 80-251-0085-5.
6. SHARP, J. Microsoft Visual C 2010. Brno: Computer Press, 2010. ISBN 978-80-251-3147-3.
7. GOYVAERTS, J, LEVITHAN, S. Regulární výrazy – Kuchařka programátora. Brno: Computer Press, 2010. ISBN 978-80-251-1935-8.
8. PRATA, S. Mistrovství v C++. Brno: Computer Press, 2004. ISBN 80-251-0098-7.

Vedoucí diplomové práce:

doc. Ing. Zdenka Prokopová, CSc.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

25. února 2011

Termín odevzdání diplomové práce:

27. května 2011

Ve Zlíně dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. RNDr. Vojtěch Křesálek, CSc.
ředitel ústavu

ABSTRAKT

Tato diplomová práce pojednává o problematice správy a údržby databázových systémů, vycházejících z technologie SQL. Téma je zaměřeno na správní a údržbový nástroj databázových systémů. Na začátku práce je vysvětlena technologie SQL, její historie vzniku a hlavní části této problematiky, jako jsou např. nástroje pro správu a údržbu RDBMS. V další části je vysvětlen moderní programovací jazyk C# a jeho využití při tvorbě aplikace, která je předmětem této práce. Hlavní část je zaměřena na popis využitých technik při programování aplikace jako nástroje pro správu, údržbu a sestavování dotazů zvoleného RDBMS. Poslední část diplomové práce obsahuje popis prostředí a práce s aplikací, která je předmětem této práce.

Klíčová slova: technologie SQL, RDBMS, jazyk C#

ABSTRACT

This thesis discusses the problem of maintaining and managing database systems based on the SQL technology. The project implements tools for management and the maintenance of database systems. At the beginning of the document, the SQL technology and its origins are explained and the main argument is introduced. The tools for managing and maintaining RDBMS are discussed. In the next part of this document, the popular programming language C# is explained in the context of the target application programming. The main part of this document is focused on various tools used for creating the application, such as tools for management, maintaining and querying the target RDBMS. The final part of the document describes the application environment and its features.

Keywords: SQL technology, RDBMS, language C#

Děkuji mému vedoucímu diplomové práce doc. Ing. Zdeňce Prokopové, CSc. za cenné rady k řešení diplomové práce a za její aktivní pomoc v průběhu celého projektu vývoje aplikace *SQLCommander*.

Dále děkuji společnosti I&C Energo a celé mojí rodině, za jejich trpělivost během mého studia.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	11
1 DATABÁZOVÉ SYSTÉMY	12
1.1 CO JE TO RDBMS	12
1.1.1 Tabulka.....	13
1.1.2 Pohled.....	13
1.1.3 Index.....	14
1.1.4 Funkce	14
1.1.5 Triger.....	15
1.2 HISTORIE JAZYKA SQL.....	16
1.3 DOTAZOVACÍ JAZYK SQL.....	17
1.4 STANDARD JAZYKA SQL	17
1.5 PRVKY JAZYKA SQL	19
1.5.1 Dotazy jazyka SQL.....	20
1.5.2 Hodnota Null a třístavové hodnoty 3VL	22
1.5.3 DML.....	23
1.5.4 DDL.....	24
1.5.5 DCL.....	25
1.5.6 Řízení transakcí.....	25
1.5.7 Komentáře jazyka SQL	26
1.6 VYLEPŠENÍ JAZYKA SQL.....	27
1.6.1 Standardní procedury jazyka SQL.....	28
1.7 NÁSTROJE PRO PRÁCI S RDBMS	29
1.7.1 Terminálové nástroje.....	30
1.7.2 Nástroje s grafickým uživatelským rozhraním.....	30
2 .NET FRAMEWORK A JAZYK C#	32
2.1 VÝZNAM PROSTŘEDÍ .NET	32
2.2 CO JE PROSTŘEDÍ .NET FRAMEWORK.....	33
2.3 VÝHODY PROSTŘEDÍ .NET	34
2.4 MODERNÍ PROGRAMOVACÍ JAZYK C#.....	36
2.5 ZÁVISLOST JAZYKA C# NA PROSTŘEDÍ .NET	39
2.6 SPOLEČNÉ PROSTŘEDÍ ZPRACOVÁNÍ	39
2.7 VÝHODY PŘEKladU DO JAZYKA IL	40
II PRAKTICKÁ ČÁST	42
3 ZÁKLADNÍ KONCEPCE APLIKACE	43
3.1 POPIS POUŽITÝCH OVLÁDACÍCH PRVKŮ	43
3.1.1 Hlavní menu	43

3.1.2	Nástrojová lišta.....	43
3.1.3	Rozložení panelů.....	44
3.1.4	SQLCommander je vícevláknová aplikace.....	44
3.1.5	Zpracování více vláken v aplikaci SQLCommander.....	45
3.1.6	TreeView.....	47
3.1.7	TabControl.....	47
3.1.8	SQLSyntaxTextBox.....	47
3.1.9	DataGridView.....	49
3.2	POPIS OVLÁDÁNÍ APLIKACE SQLCOMMANDER.....	49
3.2.1	Připojení k RDBMS.....	49
3.2.1.1	Záložka Login.....	50
3.2.1.2	Záložka Connection Specification.....	51
3.2.1.3	Záložka Expert Settings.....	52
3.2.2	Menu a nástrojové lišty.....	52
3.2.2.1	Hlavní menu aplikace.....	52
3.2.2.2	Nástrojové lišty aplikace – Toolbars.....	55
3.2.3	Levý panel aplikace.....	57
3.2.3.1	Nástrojová lišta levého panelu.....	58
3.2.4	Střední panel aplikace.....	58
3.2.4.1	Typy záložek v ovládacím prvku SQLTabControl.....	59
3.2.4.2	Editace záznamů tabulky.....	59
3.2.4.3	Formulování textu SQL dotazů.....	61
3.2.4.4	Zobrazení výstupních dat.....	62
3.2.5	Pravý panel aplikace.....	62
4	POROVNÁNÍ APLIKACE S ALTERNATIVAMI.....	63
4.1	SQL EDGE.....	63
4.2	EPICETUS.....	63
4.3	SHRNUTÍ.....	64
	ZÁVĚR.....	65
	ZÁVĚR V ANGLIČTINĚ.....	66
	SEZNAM POUŽITÉ LITERATURY.....	67
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	68
	SEZNAM OBRÁZKŮ.....	69
	SEZNAM TABULEK.....	70
	SEZNAM PŘÍLOH.....	71

ÚVOD

Problematika týkající se návrhu aplikace umožňující správu, údržbu a formulování dotazů na různé typy databázových systémů, je velmi rozsáhlá a spadá do ní celá řada jiných problematik. Dnešní nabídka aplikací poskytujících tuto funkcionalitu je stále velmi malá a většina kvalitních aplikací tohoto typu je za peníze. Při vývoji je důležité znát obecné standardy ISO SQL, protože tyto standardy se od sebe různým způsobem liší a každý výrobce databázových systémů využívá jiný standard ISO SQL. Zároveň má každý výrobce ve svém produktu implementované svoje specifické vlastnosti a funkce. Každý výrobce databázového systému také vydává nové verze, které vychází ze starší verze produktů a velmi často přidávají další funkcionalitu a vylepšení. Zde je důležité mít popis každé verze produktu daného výrobce a doplnit aplikaci pro údržbu a správu o další rozhodovací podmínky, kde se bude rozlišovat rozdíl jak mezi různými výrobci, tak mezi jejich verzemi. Toto je velmi důležité pro správnou funkci aplikace a zpětnou kompatibilitu této aplikace se všemi databázovými systémy.

Dalším významným kritériem při návrhu aplikace pro údržbu a správu databázových systémů je volba vhodného programovacího jazyka. Při výběru jsem vycházel z předpokladu, že aplikace musí být vyvíjena moderním programovacím jazykem, který je dostatečně rychlý, má dostatečnou základnu knihoven pro práci s grafickými ovládacími prvky (které tvoří rozhraní mezi uživatelem a samotným jádrem aplikace) a knihoven pro komunikaci s databázovými systémy. Dynamická knihovna, která slouží, jako rozhraní pro připojení databázového systému se odborně nazývá **Data Connector**. Tento programovací jazyk musí být navržen na objektově orientovaných základech s jednoduchou a intuitivní syntaxí zápisu kódu. S výběrem vhodného programovacího jazyka úzce souvisí ta skutečnost, že k dobrému programovacímu jazyku patří i dobré vývojové prostředí.

Nejvhodnější variantou programovacího jazyka při ohledu na všechna tato kritéria pro mě byl programovací jazyk od společnosti **Microsoft**, zvaný **C#**. Tento programovací jazyk využívá jako pracovní prostředí (pracovní rámec) **.NET Framework**. Jako vývojové prostředí jsem zvolil opět z důvodu kompatibility produkt společnosti **Microsoft**, zvaný **Visual Studio 2008**. Oba tyto produkty vyvinula společnost Microsoft, která nabízí obrovskou podporu programátorské komunitě ve formě dokumentace MSDN a dále je

programovací jazyk C#, respektive pracovního prostředí .NET Framework podporováno rozsáhlým počtem internetových konferencí (webová diskusní fóra).

Cílem této práce je vytvoření aplikace, která bude schopna provádět údržbu a správu různých databázových systémů od různých výrobců. Aplikace by měla být uživatelsky přívětivá, jak při ovládání grafického prostředí, tak by měla poskytovat dostatečné množství funkcí potřebných pro práci s databázovým systémem. Velký důraz při návrhu a vývoji této aplikace jsem také kladl na stabilitu aplikace. Posledním kritériem bylo zajištění dobré přenositelnosti aplikace na jiné verze systému Windows. Aplikace je navržena pro platformu operačního systému **Microsoft Windows**, přesněji řečeno pro platformu systému Windows podporované od verze Windows 2000 (pracovní prostředí .NET Framework je podporováno od verze 2.0).

I. TEORETICKÁ ČÁST

1 DATABÁZOVÉ SYSTÉMY

Na začátku této diplomové práce je zapotřebí zmínit, co to vlastně je ten databázový systém (databáze). Každý dnes používá databázi v běžném každodenním životě, i když o této skutečnosti ani neví. Jako příklad můžeme uvést například pokladny, kdy všechno zboží, peněžní doklady, reklamace, jsou uloženy v databázi pomocí strukturovaných tabulek a jejich relací. Dalším příkladem může být účetní systém, což je klasický příklad využití relační databáze. Databáze jsou datové sklady pro ukládání většího množství dat, u kterých požadujeme rychlost, pružnost, a možnost vytvoření více připojení v jeden okamžik. Databázové systémy mají jednu velkou výhodu a to, že je možné pracovat s daty z více klientských počítačů najednou, aniž by některý z klientů čekal na ukončení spojení předešlého klienta. Tento systém je navržen jako **vícevláknová**¹ aplikace a tak tomu i u některých databázových systémů je. Robustnější databázové systémy jsou však navrženy na systému **multiplexování**² připojení, což se jeví jako výhodnější řešení pro velké množství připojení v jeden okamžik.[1]

Moderní databázové systémy využívají k práci s daty a k dalším operacím, jako je například řízení přístupu, moderní dotazovací jazyk nazvaný **SQL**. Tento dotazovací jazyk se používá relativně dlouhou dobu a za tu dobu doznal mnoho změn a byly vytvořeny standardy, které napomáhají programátorům při komunikaci jejich aplikacemi a webovými stránkami.

1.1 Co je to RDBMS

Posledním důležitým termínem v databázových systémech je, RDBMS, což je zkratka pro databázový systém, založený na relačním modelu. Relační model znamená, že mezi

¹ Vícevláknová aplikace je aplikace, která neběží v jednom vlákne, ale dynamicky za běhu aplikace vytváří na každé spojení další vlákno, kde se zpracovávají požadavky nového připojení

² Multiplexování lze přeložit jako přepínání od prvního k poslednímu vytvořenému připojení a provedení obsluhy požadavků každého připojení

prvky databáze jsou přesně definované relace.[1] V jednodušším vyjádření tohoto významu lze říci, že mezi tabulkami v databázi existují návrhárem definované vztahy, které zajišťují integritu dat v databázi. S relacemi úzce souvisí integritní zabezpečení databáze, což lze popsat jako omezení v rámci tabulky, které je realizováno pomocí jasně definovaného typu sloupců, zdali je hodnota ve sloupci povinná či nikoli atd. Relační databáze obsahuje sadu objektů, pro ukládání, správu a přístup k datům. Tyto objekty jsou například tabulky, pohledy, indexy, funkce, trigry. Databázové objekty mohou být také definovány databázovým systémem, nebo lze vytvářet uživatelsky definované objekty.[12]

1.1.1 Tabulka

Je to základní objekt databáze, využívající se k ukládání dat ve formě řádků a sloupců. Každý řádek je jeden záznam a každý sloupec ukládá v definovaném typu jednu datovou hodnotu daného záznamu. Tabulka je definovaná svým názvem.

Příklad definice struktury tabulky jazykem SQL:

```
CREATE TABLE [users]
(
    [id] Integer IDENTITY(1,1) NOT NULL, UNIQUE ([id]),
    [name] Varchar(50) NOT NULL,
    [surname] Varchar(50) NOT NULL,
    [email] Varchar(50) NOT NULL,
    [phone] Varchar(20) DEFAULT +420 NULL,
    [enterDate] Datetime NULL,
    PRIMARY KEY ([id])
);
```

1.1.2 Pohled

Pohled, je speciální typ virtuální tabulky pro zobrazení dat v jedné nebo více tabulkách. Jinak řečeno, vytvořením pohledu a jeho zobrazením, se nám zobrazí tabulky s požadovanými daty.[16] Tyto tabulky však ve skutečnosti neexistují a jsou to pouze

dotazem vytvořené tabulky z jiných existujících tabulek databáze. Pohledy se vytvářejí v případě, že např. velmi často používáme určitý typ dotazu na zobrazení dat. V tom případě je pro nás výhodnější vytvořit si pohled, který poté zobrazíme a dostaneme stejný výsledek bez psaní zbytečně dlouhých dotazů.

Příklad definice pohledu jazykem SQL.

```
CREATE VIEW accountUser AS
-- Dotaz na všechny záznamy v tabulce accounts
SELECT * FROM accounts ORDER BY login
-- Dotaz na všechny záznamy v tabulce users
SELECT * FROM users ORDER BY id DESC;
```

1.1.3 Index

Index je speciální objekt databáze, respektive sloupce tabulky v databázi, který slouží pro rychlejší vyhledávání v rozsáhlých tabulkách, které obsahují velké množství dat. Index je obecně dobré vytvářet tam, kde víme, že tabulka bude mít opravdu hodně záznamů a hledání v záznamech by bylo opravdu pomalé, tím pádem by vrácení dat klientům na jejich dotazy trvalo příliš dlouho. Vytváření indexů nebývá častou operací, protože je za nás vytváří samotná služba RDBMS, která běží na pozadí OS.

Příklad vytvoření indexu jazykem SQL:

```
CREATE UNIQUE CLUSTERED INDEX Idx1 ON accounts(login);
```

1.1.4 Funkce

Jsou to uložené procedury databázových systémů, které jsou definované již výrobcem databázového systému, nebo je lze vytvářet a ukládat uživatelem pro pozdější využívání. Funkce jsou velmi častým objektem, který využívají především programátoři a definují je nejčastěji správci databázového systému, nebo samotní programátoři, kteří mají dostatečná oprávnění pro jejich vytváření a ukládání do databázového systému. Ve stručnosti lze říci, že funkce je určitá posloupnost SQL příkazů, které se vykonají zavoláním názvu funkce.

Jedna z možností funkce je, že dokáže vracet určitý počet a typ argumentů, které lze použít pro další zpracování. Funkce může být vytvořena pro dočasné, nebo stálé používání a zároveň lze funkci spouštět ručně, nebo ji zařadit do seznamu funkcí, které se spouští po startu SQL služby.[16]

Příklad definice funkce jazykem SQL:

```
CREATE PROCEDURE au_info_all
AS SELECT au_lname, au_fname, title, pub_name
FROM authors a INNER JOIN titleauthor ta
    ON a.au_id = ta.au_id INNER JOIN titles t
    ON t.title_id = ta.title_id INNER JOIN publishers p
    ON t.pub_id = p.pub_id;
```

Vyvolání této procedury lze provést pomocí příkazu EXECUTE nazev_funkce. Systémové funkce, které vytvořil již výrobce, mají na začátku svého názvu „sp_“, což je zkratka pro **Stored Procedure** (uložená funkce).

1.1.5 Trigger

Je to speciální typ uložené funkce, která se automaticky spouští při vzniku určité události databázového systému. Existují dva typy triggerů, které je možné vytvářet.

- **DML trigger**, jsou automaticky spouštěny při manipulaci s daty databáze pomocí příkazů INSERT, UPDATE, DELETE
- **DDL trigger**, jsou automaticky spouštěny při definici struktury databáze, která probíhá při práci s příkazy CREATE, ALTER, DROP [16]

Příklad vytvoření triggeru pro odeslání mailu při vložení, změně nebo smazání záznamu v tabulce Customer (jedná se o systémový trigger, který je vytvořený již výrobcem databázového systému):

```
CREATE TRIGGER reminder2
ON Sales.Customer
AFTER INSERT, UPDATE, DELETE
AS EXEC msdb.dbo.sp_send_dbmail
    @profile_name = 'AdventureWorks2008R2
Administrator',
    @recipients = 'danw@Adventure-Works.com',
```

```
@body = 'Don''t forget to print a report for the  
sales force.',  
@subject = 'Reminder';
```

1.2 Historie jazyka SQL

V sedmdesátých letech minulého století vyvinula skupina **IBM San Jose Research Laboratories** relační databázový systém nazvaný **System R**, založený na modelu uvedeném *Edgarem F. Coddem* v jeho zprávě *A Relation Model of Data for Large Shared Data Banks*³. *Donald D. Chamberlin* a *Raymond F. Boyce* ze společnosti IBM následně vyvinuli **SEQUEL** - Structured English Query Language, pro manipulaci a řízení dat v jejich relačním databázovém systému System R. Výraz SEQUEL byl později přejmenován na SQL, protože výraz SEQUEL byl obchodní značkou anglické letecké společnosti s názvem *UK-based Hawker Siddeley*. Koncem sedmdesátých let minulého století, společnost *Relation software, Inc.*, nyní **Oracle Corporation** vyvinula svůj vlastní relační databázový systém a usilovala o prodej tohoto RDBMS americké armádě, centrální agendě rozvědky a ostatním americkým vládním institucím. V létě roku 1979 *Relation Software, Inc.* představilo první komerčně dostupný produkt implementace SQL, nazvaný **Oracle V2** pro počítače VAX. Tento produkt urychlil vydání produktu **System/38**, společnosti IBM.[13]

Po otestování technologie SQL na straně zákazníka, kde se potvrdil obrovský potenciál RDBMS, společnost IBM začala vyvíjet komerční produkty založené na jejich System R a SQL/38 s názvy SQL/DS a DB2, které byly komerčně dostupné v letech 1979, 1981, respektive 1983.[13]

³ Kompletní zpráva je dostupná na adrese <http://www.scribd.com/doc/14498590/A-Relational-Model-of-Data-for-Large-Shared-Data-Banks>

1.3 Dotazovací jazyk SQL

SQL (Structured Query Language) je databázový dotazovací jazyk, navržen pro získávání a údržbu dat v relačních databázových systémech (RDBMS), modifikaci nebo vytváření databázových schémat (struktur tabulek), vytváření a řízení přístupových úrovní k jednotlivým objektům databázového systému.

SQL bylo poprvé standardizováno společností ANSI a následně společností ISO. Většina databázových systémů implementuje tyto standardy a často přidává svoje vlastní vylepšení. SQL poskytuje funkce pro vkládání, aktualizaci nebo mazání dat. Databázový údržbový systém také poskytuje administrátorské nástroje a diagnostické nástroje. Většina databázových systémů také poskytuje **rozhraní příkazové řádky (SQL/CLI)**, kde je možné zadávat dotazy pro vkládání, aktualizaci nebo mazání záznamů. Lze provádět zálohování databází, změnu práv k daným databázovým objektům. SQL/CLI je negrafické rozhraní a není uživatelsky tak přívětivé jako grafické prostředí, které poskytují **GUI**.

První verze jazyka SQL byla vyvinuta americkou společností IBM, pány *Andrew Richardsonem*, *Donaldem C. Messerlym* a *Raymondem F. Boycem*, na počátku sedmdesátých let minulého století, měla název **SEQUEL** a sloužila výhradně pro ukládání, aktualizaci a mazání dat v relační databázi System R. Společnost IBM si nechala tuto první verzi relačního databázového systému patentovat v roce 1985. V roce 1986 byl následně vytvořen standard SQL-86 asociací ANSI, pro mezinárodní standardizaci.[13]

Původní návrh SQL jazyka byl definovat deklarativní dotazy a jazyk pro práci s daty. Výrobci RDBMS začínají přidávat svoje uživatelsky definovatelné datové typy a mnoho dalších vylepšení. S vydáním standardu SQL v roce 1999 byla velká část vylepšení výrobců formálně přijata jako součást SQL jazyka. Tyto vylepšení jsou zahrnuty v **Persistent Stored Modules (SQL/PSM)**.

1.4 Standard jazyka SQL

SQL byl přijat jako standard asociací ANSI v roce 1986 a asociací ISO v roce 1987. V původním SQL standardu, asociace ANSI deklarovala, že oficiální název je SQL,

vyslovováno *es kjú el*. Avšak mnoho anglicky mluvících databázových profesionálů stále používalo výraz SEQUEL, vyslovováno *sikvel*. SEQUEL byl vyvinut ve společnosti IBM a byl původcem zkratky dnešního SQL.[2]

V průběhu roku 1996, byl mezinárodní institut standardů a technologií (NIST) požádán o vydání standardu, který definuje SQL. Od této chvíle musí každý výrobce relačního databázového systému po vydání nové verze produktu definovat, jakému standardu daný produkt vyhovuje⁴.

SQL Standard není volně dostupný, ale je možné jej zakoupit od společnosti ISO nebo asociace ANSI. Mimochodem návrh SQL 2008 je volně ke stažení jako ZIP soubor.

Tabulka 1: Seznam SQL standardů. Zdroj [13]

Rok	Jméno	Alias	Poznámka
1986	SQL-86	SQL-87	Poprvé vydán asociací ANSI a ohodnocen společností ISO
1989	SQL 89	FIPS 127-1	Vedlejší revize, přijata jako FIPS 127-1
1992	SQL-92	SQL2, FIPS 127-2	Vedlejší revize (ISO9075), vstupní úroveň SQL-92, přijata jako FIPS 127-2
1999	SQL:1999	SQL3	Přidány porovnávání regulárních výrazů, Rekurzivní dotazy, trigry, podpora procedurálních a "Control-of-Flow" zápisů, neskálární typy, a několik funkcionalit, týkajících se objektově orientovaných rysů
2003	SQL:2003	SQL 2003	Představeno XML-relační rysy, funkce okna, standardizované sekvence, a sloupce s automaticky generovanými hodnotami

⁴ Kompletní popis standardů ISO SQL je dostupný z <http://www.iso.org>

			(sloupeček typu indenty)
2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006 definuje jakou cestou může být SQL použito ve spojení s XML. Definuje cesty jak importovat a ukládat XML data v SQL databázích
2008	SQL:2008	SQL 2008	Legalizován výraz ORDER BY vně definic kurzoru. Přidány INSTEAD OF trigry. Přidán TRUNCATE zápis

1.5 Prvky jazyka SQL

Jazyk SQL je rozdělen do několika částí, zahrnujících:

- Klauzule** - Jsou v některých případech doporučenými komponenty zápisu⁵
- Výrazy** - Mohou produkovat skalární⁶ hodnoty, nebo tabulky, zahrnující jejich sloupce a řádky jako jejich data
- Predikáty** - Předvídají, které specifické podmínky mohou být hodnoceny jako *True value logic* – 3VL⁷
- Dotazy** - Jsou používány pro získávání, aktualizaci nebo mazání dat, řízení přístupových práv, transakcí, dotazy jsou nejzákladnějším prvkem jazyka SQL [13]

⁵ Dotaz je v tomto kontextu myšleno SQL dotaz

⁶ Skalární typ proměnné je základní datový typ, což je takový, který si dokáže v jednu chvíli zapamatovat pouze jednu hodnotu, např. typ INT, CHAR, BYTE

⁷ 3VL je typ proměnné, která může nabývat tři hodnot, např. proměnná typu BOOL nabývá hodnot TRUE, FALSE, Nedefinováno

Jazyk SQL obsahuje také speciální znaky, které mu říkají, kde je konec dotazu a začíná další dotaz. K tomuto účelu se používají znak středník “;”, který oznamuje překladači SQL dotazu konec dotazu a jako druhý oddělovač se používá příkaz “GO”, který oznamuje překladači SQL jazyka, že v tomto místě končí dávkový příkaz⁸.

Jazyk SQL zpracovává také speciálním způsobem tzv. “bílé znaky”. Bílé znaky jsou v podstatě ignorovány překladačem SQL jazyka a to z důvodu vyššího komfortu formulování SQL dotazů. Jakým způsobem zpracovává oddělovací znaky a bílé znaky překladač SQL jazyka, záleží na zvolené aplikaci, ve které se provádí formulování textu SQL dotazů. Každý výrobce má ve své aplikaci menší odlišnosti.[3]

1.5.1 Dotazy jazyka SQL

Nejběžnější operace vykonávané v SQL databázích jsou práce s dotazy, které se provádí deklarací klíčového slova **SELECT**. SELECT vrací data z dotazované tabulky, data z více tabulek spojené relacemi nebo data funkcí a výrazů. Příkaz SELECT nespadá do **DML** výrazů, protože nemá přímo vliv na data uložená v databázi. Je však několik platformě specifických variací příkazu SELECT, které mohou mít vliv na uložená data v databázi, jako například zápis SELECT INTO, který je použit v některých RDBMS. SQL dotaz zahrnuje seznam sloupců, který následuje ihned za klíčovým slovem SELECT. Na místo seznamu sloupců lze také použít hvězdičku “*”, jako zástupný znak, který vrátí všechny sloupce tabulky, nebo tabulek pokud je dotaz volán na více tabulek. Příkaz SELECT je nejkompaktnější výraz v SQL jazyce, který zahrnuje mnoho dalších klíčových slov a parametrů.

- Klauzule **FROM**, která definuje zdrojovou tabulku, nebo zdrojové tabulky, ze kterých budou načítána data. Klauzule FROM může obsahovat další klíčové

⁸ Dávkové příkazy nepodporují všichni výrobci RDBMS a jedná se zpracování více SQL příkazů najednou

slovo **JOIN** ke spojení dalších tabulek, založených na uživatelsky definovaných kritériích.

- Klauzule **WHERE** zahrnuje porovnávací predikát, který je použit k omezení počtu vrácených záznamů daného dotazu. Klauzule **WHERE** je použita před klauzulí **GROUP BY**. Klauzule **WHERE** eliminuje všechny záznamy z výsledného datasetu⁹, u kterých se porovnání podmínek **WHERE** nerovná *True*¹⁰.
- Klauzule **GROUP BY** je používána ke kombinaci nebo seskupování řádků s relačními daty. Klauzule **GROUP BY** je často používána ve spojení s agregačními funkcemi jazyka **SQL**, nebo pro odstranění duplicitních řádků z výsledného datasetu
- Klauzule **HAVING** se používá, k odstranění řádků po použití klauzule **GROUP BY** na výsledný dataset.
- Klauzule **ORDER BY** se používá pro definování toho, který sloupec nebo sloupce budou použity pro seřazení výsledného datasetu. Klauzule **ORDER BY** se používá, protože vrácený dataset je vrácen neseřazený a nelze předvídat jak bude vrácený dataset zobrazen. Obvykle se doplňuje klauzulí **ORDER BY** a klauzulí **ASC** nebo **DESC**, což definuje sestupné nebo vzestupné řazení dat.[13]

Následující příklad provede dotaz na záznamy z tabulky *satelity* a výsledný dataset vrátí všechny záznamy z dané tabulky, kde je *cena* nižší než 4000. Výsledný dataset bude poté seřazen sestupně. Hvězdička za klauzulí **SELECT** specifikuje zobrazení všech sloupců z dané tabulky.

```
SELECT * FROM satelity WHERE cena < 4000;
```

⁹ Dataset je seznam požadovaných záznamů z tabulky, nebo tabulek, specifikované dotazem **SELECT**

¹⁰ Hodnota *True* je stav, kdy daný výraz je pravdivý, naopak hodnota *False* nastane, pokud daný výraz je nepravdivý

Další příklad demonstruje vícenásobné spojení tabulek, klauzulí JOIN a GROUP. Tento dotaz vrátí seznam všech jmen satelitů z tabulky *satelity* a počet výrobců satelitů z tabulky *vyrobci*.

```
SELECT satelity.Nazev, count(*) AS Vyrobci
FROM satelity JOIN vyrobci ON satelity.vyrobce id =
vyrobci.id
GROUP BY satelity.Nazev;
```

Výstup tohoto dotazu znázorňuje Tabulka 2.

Tabulka 2: Výstup SQL dotazu počtu satelitů. Zdroj [Vlastní zpracování]

Nazev	Vyrobci
DreamBox DM7025	3
DreamBox DM500	5

1.5.2 Hodnota Null a třístavové hodnoty 3VL

Myšlenka zavedení hodnoty **Null** do jazyka SQL byla realizována z důvodu zachycení chybějící nebo nezadané hodnoty v relační databázi. Z toho vyplývá, že pokud nezadáme hodnotu do některého pole při vkládání nového záznamu, nebude toto pole obsahovat žádnou hodnotu a tím pádem bude mít automaticky hodnotu Null. Speciálním případem je nevložení hodnoty do pole typu BOOL. V takovém případě nebude mít pole hodnotu False, ale hodnotu Null, s čímž souvisí třístavové hodnoty proměnných. Slovo Null je zároveň klíčové slovo jazyka SQL, použité pro definování speciálního znaku Null.[11]

V jazyce SQL lze využít dvou speciálních výrazů pro definování, nebo zjištění, zdali pole nabývá hodnoty Null nebo ne.

IS NULL - Definuje, nebo zjišťuje, zdali pole obsahuje hodnotu Null

IS NOT NULL - Definuje, nebo zjišťuje, zdali pole neobsahuje hodnotu Null

1.5.3 DML

Data Manipulation language, neboli jazyk pro změnu dat, je část příkazů jazyka SQL pro přidávání, změnu a mazání dat v relačních databázích.[11]

- **INSERT** je používán pro přidávání nových záznamů do tabulky relační databáze

```
INSERT INTO tabulka1 (Sloupec1, Sloupec2, Sloupec3)
VALUES (1, 'Název', 26.4);
```

- **UPDATE** se používá pro změnu dat v tabulce, nebo změnu struktury tabulky nebo databáze. Následující příklad provede změnu hodnot jednoho záznamu v tabulce *tabulka1*.

```
UPDATE tabulka1
SET Sloupec1 = 1, Sloupec2 = 'Název', Sloupec3 = 26.4
WHERE id = 6;
```

- **DELETE** smaže 0 nebo více záznamů z tabulky, což je závislé na klauzuli WHERE, nebo smaže celý objekt z RDBMS, což může být např. tabulka nebo databáze. Následující příklad smaže z tabulky *tabulka1* záznam s *id* = 48 a *vyrobce* = 'Dream multimedia'.

```
DELETE FROM tabulka1
WHERE id = 48 AND vyrobce = 'Dream multimedia';
```

- **MERGE** se používá pro kombinaci dat z více tabulek. Je to jakási kombinace klauzulí INSERT a UPDATE. Klauzule MERGE byla přidána SQL standardu *SQL:2003*. Mnoho výrobců databázových systémů poskytuje podobnou funkci, která se liší pouze zápisem a velmi často se nazývá **UPSERT**.

1.5.4 DDL

Data definition language, neboli jazyk pro definici dat, slouží pro návrh nových prvků databázového systému, změnu nebo odstranění. Většina komerčních SQL databází má DDL s proprietárními vylepšeními, které umožňují kontrolu nad nestandardními prvky databázového systému. Mezi základní prvky DDL patří příkazy CREATE, ALTER, RENAME, TRUNCATE a DROP.[13]

- CREATE je nejčastěji používán pro vytvoření nové databáze, tabulky, indexu nebo relace mezi tabulkami
- DROP způsobí, že název objektu předaný tomuto příkazu je nevratně odstraněn, například tabulka nebo databáze
- TRUNCATE smaže všechny data z tabulky (rychlý způsob mazání záznamů v tabulce)
- ALTER umožňuje uživateli měnit existující objekty databáze. Tímto příkazem lze např. přidávat nové sloupce do tabulky, nebo měnit parametry existujících sloupečků

Následující příklad DDL vytvoří novou tabulku *tabulka1* se třemi sloupečky různého typu a definuje primární klíč¹¹ ze sloupců *Sloupec1* a *Sloupec2*. Sloupce *Sloupec1* a *Sloupec2* jsou povinné, tzn., že při vkládání nového záznamu do *tabulky1* je zapotřebí vyplnit tyto pole. Naopak *Sloupec3* je nepovinný a pokud nevyplníme jeho hodnotu při vkládání nového záznamu do tabulky, bude nabývat hodnoty Null.

```
CREATE TABLE tabulka1
(
    Sloupec1 INT NOT NULL,
    Sloupec2 VARCHAR(50) NOT NULL,
```

¹¹ Primární klíč tabulky má vždy unikátní hodnotu ve svém poli a používá se jako unikátní identifikátor záznamu v dané tabulce

```
Sloupec3 DATETIME NULL,  
PRIMARY KEY (Sloupec1, Sloupec2)  
);
```

1.5.5 DCL

Data control language, neboli jazyk pro řízení přístupu (k datům nebo objektům RDBMS), je třetí část jazyka SQL, který slouží k definování a řízení přístupových práv v databázových systémech. DCL se skládá ze dvou klíčových slov a to:

- **GRANT** povoluje jednomu, nebo více uživatelům vykonávat operaci nebo skupinu operací nad daným objektem, na který byl příkaz GRANT použit
- **REVOKE** odebírá, nebo zakazuje přístup uživateli vykonávat operaci, nebo skupinu operací nad daným objektem, na který byl příkaz REVOKE použit [11]

Příklad povolení operací SELECT a UPDATE na tabulku s názvem *tabulka1* a odebrání operací SELECT a UPDATE na tabulce s názvem *tabulka2*.

```
GRANT SELECT, UPDATE  
ON tabulka1  
TO uzivatel1, uzivatel2;  
  
REVOKE SELECT, UPDATE  
ON tabulka2  
TO uzivatel1, uzivatel2;
```

1.5.6 Řízení transakcí

V případě, že databázový systém podporuje transakce, lze „obalit“ DML operace transakcemi. Transakce slouží k tomu, aby bylo možné změny provedené na datech databáze vrátit na původní hodnoty. Transakce jsou velmi důležitá funkcionalita jazyka SQL, která dodává databázovým systémům velkou komfortnost a zajišťuje bezpečnost při práci s důležitými daty.

- **START TRANSACTION** (nebo také **BEGIN WORK**, nebo **BEGIN TRANSACTION**, záleží na SQL dialektu), může být použito pro označení začátku transakcí.
- **COMMIT** tato klauzule způsobí, že změny provedené na datech budou trvalé
- **ROLLBACK** tato klauzule způsobí, že všechny změny na datech provedené do posledního příkazu **COMMIT** nebo **ROLLBACK** budou vrácena zpět na původní hodnoty

Poté, co je zavolán příkaz **COMMIT**, není možné vrátit zpět původní data. Jinak řečeno, po zavolání příkazu **COMMIT** není možné zavolat opět příkaz **ROLLBACK** pro vrácení se k původním hodnotám záznamů. Příkazy **COMMIT** a **ROLLBACK** jsou závislé na daném typu SQL standardu a vydavateli DBMS a jejich zápis se může lišit.[2]

Následující příklad zobrazuje klasické použití transakcí v bankovních RDBMS.

```
START TRANSACTION;  
UPDATE ucet SET celkova_castka = celkova_castka + 36520.5  
WHERE cislo_uctu = 25489;  
UPDATE ucet SET celkova_castka = celkova_castka - 3000  
WHERE cislo_uctu = 23289;  
IF ERROR = 0 COMMIT;  
IF ERROR <> 0 ROLLBACK;
```

1.5.7 Komentáře jazyka SQL

Mezi ostatní lze zahrnout řídicí znaky pro jednořádkové a víceřádkové komentáře, které definuje standard ISO pro jazyk SQL. Pro definování jednořádkového komentáře se v jazyce SQL používají dva po sobě jdoucí znaky pomlčky "--" nebo znak mřížky "#". Po zjištění tohoto řídicího znaku se ignoruje zbylá část znaků na daném řádku a začátek dalšího dotazu se zahájí na dalším řádku. Pro definici víceřádkového komentáře se používají zahajovací a ukončovací znaky "/*" a "*/".

Příklad jednořádkového a víceřádkového komentáře jazyka SQL:

```

-- Dotaz s podmínkou WHERE
SELECT * FROM feeValues WHERE (fee >= 500) AND (fee <=
10000)

/*
Created          21.3.2010
Modified         19.8.2010
Project          Satelites
Model            MainSatModel
Company          UTB FAI Zlín
Author           JNOVAK
Version          1.0
Database         MS SQL 2000
*/
create database satelites

```

1.6 Vylepšení jazyka SQL

SQL technologie je navržena pro specifický účel: Dotazovat se na data umístěná v relačních databázových systémech. SQL technologie je založena na *Datasetech* a je základem naprosto odlišná od programovacích jazyků jako jsou C/C++ nebo C#. Nicméně byly přidány do SQL standardu nastavby, které přidávají funkcionalitu procedurálního programovacího jazyku, jako například konstrukce "Control-of-Flow", viz Tabulka 3¹².

Tabulka 3: Seznam procedurálních vylepšení jazyka SQL. Zdroj [13]

Zdroj	Společný název	Plné jméno
ANSI/ISO standard	SQL/PSM	SQL/Persistent Stored Procedures
IBM	SQLPL	Procedural SQL
Microsoft/Sybase	T-SQL	Transact-SQL
MySQL	SQL/PSM	SQL/Persistent Stored Module (implementuje

¹² Detailní popis procedurálních vylepšení je dostupný z <http://www.iso.org>

		SQL/PSM)
Oracle	PL/SQL	Procedural Language/SQL (založeno na ADA)
PostgreSQL	PL/PgSQL	Procedural Language/Persistent Stored Modules (implementuje SQL/PSM)
PostgreSQL	PL/SQL	Procedural Language/Persistent Stored Modules (implementuje SQL/PSM)

1.6.1 Standardní procedury jazyka SQL

SQL standard je rozdělen na několik základních částí, které zahrnují:

- *SQL/Foundation*, je definované v ISO/IEC 9075, část 2. Tato část standardu obsahuje většinu centrálních elementů SQL jazyka. Sestává jak z povinných, tak z doporučených prvků jazyka SQL.
- *SQL/CLI*, nebo také *Call Level Interface* je definovaná v ISO/IEC 9075, část 3. Definuje společné vstupní komponenty (struktury a komponenty), které mohou být použity k vykonání SQL dotazů z jiných programovacích jazyků. SQL/CLI je definováno jako nezávislý běh dalšího vlákna aplikace, ve kterém jsou vykonávány SQL výrazy a procedury, odděleně od zdrojových kódů hlavní aplikace
- *SQL/PSM*, nebo také *Persistent Stored Modules* je definovaná v ISO/IEC 9075, část 4. Standardizuje procedurální nástavby SQL, zahrnuje toky řízení, udržování stavů. SQL/PSM specifikuje deklaraci a údržbu rutin databázového jazyka (Stored Procedures)¹³. Tato část standardu obsahuje doporučené prvky SQL.
- *SQL/MED*, nebo také *Management of External Data* je definovaná v ISO/IEC 9075, část 9. Tato část definuje práci s externími daty. Externí data jsou taková, která jsou

¹³ Stored Procedures jsou doslova uložené procedury, nebo funkce, které velmi často vykonávají několik SQL dotazů dohromady a uživatel vyvolá pouze název uložené procedury

dostupná, ale nejsou řízena DBMS založené na SQL. Tato část standardu obsahuje doporučené prvky SQL.

- *SQL/OLB*, nebo také *Object Language Bindings* je definovaná v ISO/IEC 9075, část 10. SQL/OLB definuje zápis a sémantiku SQLJ, což je typ SQL dlouhá léta využívaný v programovacích jazycích Java¹⁴. Tato část standardu obsahuje doporučené prvky SQL.
- *SQL/Schemas*, nebo také *Information and Definition Schemas* je definovaná v ISO/IEC 9075, část 11. SQL/Schemas definuje informační schéma a definiční schéma, která poskytují společné nástroje k tomu, že databáze a jejich objekty jsou „samopopisující se“. Tato část standardu obsahuje doporučené prvky SQL.
- *SQL/JRT*, nebo také *SQL Routines and Types for the Java Programming Language* je definovaná v ISO/IEC 9075, část 13. SQL/JRT specifikuje schopnost volat statické Java metody a obvyklé postupy přímo mezi Javou a SQL aplikacemi. Dále je zde specifikována možnost použití Java tříd jako strukturovaných datových typů pro SQL. Tato část standardu obsahuje doporučené prvky SQL.
- *SQL/XML*, nebo také *XML Related Specifications* je definovaná v ISO/IEC 9075, část 14. SQL/XML specifikuje SQL vylepšení pro využití XML ve spojení s SQL. Je poprvé představen datový typ XML, jsou přidány nové funkce pro práci s datovým typem XML. Tato část standardu obsahuje doporučené prvky SQL.[13]

1.7 Nástroje pro práci s RDBMS

Každý výrobce relačního databázového systému poskytuje ke svému produktu také nástroj, kterým lze pracovat s jeho RDBMS. Tyto nástroje lze rozdělit do dvou základních kategorií:

¹⁴ Java je objektově orientovaný programovací jazyk, založený na základech programovacího jazyka C++. Tento programovací jazyk je multiplatformní a vyvinula jej firma SUN Microsystems

1.7.1 Terminálové nástroje

Jsou softwarové nástroje bez podpory grafického rozhraní. Veškerá údržba a správa RDBMS se provádí z příkazové řádky konzoly, prostřednictvím psaní SQL příkazů do řádků. Tento nástroj poskytuje každý výrobce RDBMS, ale je uživatelsky velmi nepřívětivý a neposkytuje žádnou větší flexibilitu ani komfort při formulování textu SQL dotazů.

1.7.2 Nástroje s grafickým uživatelským rozhraním

Jsou aplikace s podporou grafického rozhraní GUI. Veškerá údržba a správa RDBMS se provádí přes grafické rozhraní, což je prezentováno jako formuláře a pohledy nativní¹⁵ aplikace pro danou platformu, nebo se prezentuje prostřednictvím *webového prohlížeče*, kde jsou jako grafické rozhraní použity také formuláře a pohledy.

Grafické nástroje pro údržbu databází ovšem neposkytují všichni výrobci RDBMS a proto byly tyto grafické nástroje vyvinuty jinými společnostmi, zabývajícími se vývojem softwaru. Většina těchto aplikací má však podporu pouze vybraných RDBMS a většina těchto aplikací není volně dostupná. Aplikace, která je předmětem této diplomové práce je volně šiřitelná.

Problematika zabývající se tvorbou aplikace s grafickým uživatelským rozhraním, která je schopna spravovat více DBMS, je velmi rozsáhlá. Hlavní úskalí této problematiky je v dodržování všech dnes vydaných standardů SQL jazyka. Dalším problémem je to, že každý výrobce RDBMS má svoje specifické příkazy a konstrukce výrazů, které se dále mohou lišit v každé nově vydané verzi. Z těchto problémů poté samozřejmě ústí problém, že grafické rozhraní takovéto aplikace bude pro každého výrobce RDBMS a pro každou jeho verzi jiné a tím se zvyšuje složitost a pracnost aplikace. S velkou pracností úzce

¹⁵ Je to aplikace, která je přeložena přímo pro tuto platformu a nevyžaduje další software ke spuštění

souvisí vyšší časová náročnost vývoje aplikace a její vyšší cena, pokud se budeme pohybovat na hladině placených aplikací.

Aplikace *SQLCommander*, v současné verzi podporuje většinu standardů MsSQL a MySQL, které jsou jedny z nejběžněji používanými RDBMS dnes. Při vývoji této aplikace byl kladen velký důraz na přívětivost a intuitivnost grafického uživatelského rozhraní, k čemuž také velmi přispívá systém záložek, ve kterých si uživatel zpracovává nové SQL dotazy nebo edituje data v tabulkách, ale především systém barevného zvýrazňování SQL dotazů, při jejich formulaci.

2 .NET FRAMEWORK A JAZYK C#

2.1 Význam prostředí .NET

Chceme-li porozumět významu prostředí .NET, musíme si uvědomit povahu mnoha technologií určených pro systém Windows¹⁶, které se na trhu objevily za posledních deset let. Přestože se to může jevit jinak, mají všechny operační systémy Windows, verzí Windows 3.1 (uvedenou v roce 1992) počínaje a verzí Windows 7 konče, ve svém jádru stejné rozhraní pro programování aplikací (API). Nové verze Windows pouze rozšiřují stávající rozhraní API o nové funkce. Tento proces, ale spíše rozšiřuje stávající rozhraní, než že by jej nahrazoval.

To samé lze říci i o mnoha dalších technologiích a pracovních prostředích používaných pro vývoj aplikací pro operační systémy Windows. Například programovací model **COM** vznikl původně jako technologie **OLE**. Tehdy to však byl v podstatě prostředek propojení různých typů kancelářských dokumentů. Prostředí .NET je technologie vyvinutá společností Microsoft a pro vývoj této technologie se zmíněná společnost rozhodla proto, že jí šlo o zpětnou kompatibilitu. Po celá léta bylo pro operační systém Windows dalšími společnostmi vyvinuto obrovské množství aplikací a tento systém by nikdy nedosáhl takového úspěchu, kdyby společnost Microsoft zavedením nové technologie přetrhala spojení s existujícími knihovnamí a funkcemi již používaných operačních systémů a jejich technologií.[5]

I když je zpětná kompatibilita jednou z klíčových vlastností technologií určených pro operační systém Windows, je to rovněž jedna z jeho velkých slabín. Každá nová technologie je rozvinutím technologie předchozí a jako taková přináší nové funkce a možnosti. Výsledkem je vždy o něco komplikovanější produkt, který je především závislý na funkcionalitě a knihovnách verze předešlé.

Prostředí .NET je jednou z těchto nových technologií na kterou vynaložila společnost Microsoft nemalé finanční i časové prostředky. Startem do nové éry je právě jazyk C#

¹⁶ Windows je operační systém firmy Microsoft s grafickým uživatelským prostředím

a pracovní prostředí .NET. Co je to .NET ? Je to pracovní a operační prostředí, nové rozhraní pro programování aplikací (API) v systému Windows. Jazyk C# je novým programovacím jazykem, který je vytvořen od základu nový a je navržen výhradně pro programování aplikací pro prostředí .NET. Tento jazyk umožňuje využití všech výhod vývojářského prostředí a lepší pochopení základů objektově orientovaného programování (OOP), s nímž se setkáváme dobrých dvacet let.

Zde je důležité zdůraznit, že při tvorbě operačního prostředí .NET nedošlo ke ztrátě zpětné kompatibility. Existující programy budou fungovat i nadále a prostředí .NET bylo navrženo tak, aby umožňovalo spouštění také starších programů. Komunikaci mezi softwarovými komponentami v systému Windows, zajišťuje nyní téměř výhradně prostředí COM. Tento fakt nelze přehlédnout a proto nabízí prostředí .NET zapouzdření existujících komponent COM tak, aby s nimi mohli úspěšně komunikovat i přirozené komponenty modelu .NET.[5]

2.2 Co je prostředí .NET Framework

Nejlépe na tuto otázku odpovím přirovnáním prostředí .NET k systému Windows a otázkou „čím je vlastně Windows pro vývojáře ?“. Odpověď je dvojitá: Systém Windows je především knihovnou. Je to množina všech funkcí v rozhraní Windows API, dostupných při programování aplikací. Zmiňované aplikace nabízejí společné rysy - zobrazením dialogů, čí rozhraním pro více dokumentů **MDI** počínaje a přístupem k základním funkcím, jako jsou zabezpečení a služby komponent konče. Zadruhé je systém Windows prostředím, v němž jsou vaše aplikace spouštěny. Kromě toho je to také operační systém¹⁷.

Stejně tak lze rozdělit i prostředí .NET. Zprv je to knihovna, která je stejně rozsáhlá a kompletní jako Windows API. Můžeme ji používat k volání stejných funkcí, které dříve poskytoval samotný operační systém Windows: zobrazení dialogů, ověření uživatelských pověření, volání základních služeb operačního systému, nebo tvorba nových vláken.

¹⁷ Kompletní popis .NET je dostupný na <http://www.microsoft.com>

Kromě toho můžeme tuto knihovnu využívat i v novějších oblastech, jako je přístup k databázím nebo připojení k internetu.[15]

Prostředí .NET nabízí rovněž operační prostředí (.NET Runtime), v němž jsou programy spouštěny. Při spouštění kódu založeného na platformě .NET to bude právě operační prostředí .NET, které náš kód spustí. A nejen to, rovněž zajistí správu spuštěných vláken, podpůrných služeb a v pravém slova smyslu prvotní prostředí, s nímž se náš kód setká. Na prostředí .NET se tedy můžeme dívat jako na něco, co zobecňuje operační systém.

2.3 Výhody prostředí .NET

Již jsem zmínil, jak je prostředí .NET skvělé, ještě jsem se však nezmínil o tom, jak může usnadnit život vývojáři. Následuje seznam zdokonalených rysů popisovaného prostředí.

- **Objektově orientované programování.** Prostředí .NET a jazyk C# jsou od svého počátku postaveny na důsledném dodržování objektově orientovaného programování, na kterých jsou postaveny jejich základy.
- **Nezávislost jazyka.** V prostředí .NET, je kód všech jazyků (VB.NET, C#, J# a řízeného C++) překládán do *společného kódu*. To znamená, že programy v různých jazycích mohou spolupracovat dosud nevídaným způsobem.
- **Efektivní přístup k datům.** Množina komponent prostředí .NET souhrnně označována jako **ADO.NET**, poskytuje efektivní přístup k relačním databázím a jiným datovým zdrojům. Kromě toho je do všech tříd zabudována podpora technologie **XML**, což umožňuje import nebo export dat i mezi systémy, které nejsou postaveny na platformě Windows.
- **Sdílení kódu.** V prostředí .NET byl zcela revidován způsob sdílení kódu mezi aplikacemi. Byl zaveden pojem *sestavení* (nebo také seskupení, assembly), který nahrazuje tradiční dynamickou knihovnu DLL. Sestavení mají formální možnosti sledování verzí. Systém může obsahovat více verzí sestavení.
- **Vylepšené zabezpečení.** Každé sestavení může obsahovat integrovanou bezpečnostní informaci, jejíž prostřednictvím lze bezpečně vyjádřit, kdo může volat

metody jednotlivých tříd. To poskytuje velmi slušnou úroveň kontroly nad způsobem nasazení sestavení.

- **Instalace s nulovým účinkem.** Existují dva druhy sestavení (assembly): *sdílené* a *soukromé*. Sdílená sestavení jsou společnými knihovny dostupnými veškerému softwaru. Soukromá sestavení jsou určena k využití pouze určitého softwaru. Soukromé sestavení je zcela soběstačné, takže proces jeho instalace je velmi jednoduchý, protože nevyžaduje žádné záznamy v systémovém registru. Příslušné soubory jsou jednoduše vloženy do odpovídající složky v souborovém systému daného počítače.
- **Podpora webových služeb.** Prostředí .NET obsahuje zcela integrovanou podporu vývoje a tvorby webových služeb. Webové služby lze nyní vyvíjet stejně snadno jako jakýkoli jiný typ aplikací.
- **Visual Studio .NET.** Prostředí .NET má nové vývojové prostředí nazvané *Visual Studio .NET*. V něm můžeme psát kód v jazycích C++, C#, VB.NET, ale i kód webových stránek ASP.NET.
- **Jazyk C#.** Je nový zcela objektově orientovaný programovací jazyk určený k programování v prostředí .NET.



Obrázek 1: Logo pracovního prostředí .NET Framework, Zdroj [15]

2.4 Moderní programovací jazyk C#

V určitém smyslu lze jazyk C# považovat za totéž, čím je platforma .NET pro prostředí Windows. Stejně jako se za posledních deset let rozšířilo rozhraní Windows API, rozšířily se i jazyky Visual Basic a C++. Přestože se nakonec z obou jazyků v důsledku všech změn staly poměrně velmi výkonné nástroje, sužovaly je problémy zděděné z předchozích verzí.[5]

V případě jazyka Visual Basic je hlavní silou jazyka nejen nesmírná jednoduchost jeho syntaxe, ale i jednoduchost provedení požadovaných úloh. Vývojář prakticky nemusí znát žádné podrobnosti základního rozhraní API ani infrastrukturu komponent modelu COM. Nedostatkem jazyka Visual Basic je skutečnost, že nikdy nebyl doopravdy objektově orientovaný. Nepříjemným důsledkem tohoto faktu je chaotický kód a poměrně složitá údržba rozsáhlých aplikací. Syntaxe jazyka Visual Basic pochází z dob ranních verzí jazyka BASIC (který byl navržen, aby jej mohli používat začínající programátoři, nikoli proto, aby v něm bylo možné psát rozsáhlé komerční aplikace). Není tedy vhodný pro tvorbu správně strukturovaných nebo objektově orientovaných aplikací.

Pak je tu jazyk C++, jehož kořeny spadají do definice standardu ISO C++. S tímto standardem není zcela kompatibilní a to z jednoduchého důvodu. Programátoři ve společnosti Microsoft napsali svůj první překladač jazyka C++ ještě dříve, než byla definice ISO C++ uznána za oficiální. Rozdíly nejsou velké, vedly ovšem ke dvěma základním problémům. Jazyk ISO C++ má své kořeny v technologii více než deset let staré a stále výrazněji se ukazuje, že neobsahuje podporu pro moderní pojmy (například řetězce *Unicode*¹⁸ a tvorbu dokumentů XML). Obsahuje rovněž určité archaické syntaktické struktury navržené pro překladače minulosti (například oddělení deklarace od definice členských funkcí). Zadruhé, společnost Microsoft se současně snažila vyvinout jazyk C++ jako jazyk, jenž by byl vhodný pro programování velmi výkonných úloh, spouštěných v systému Windows. Kvůli tomu musel tento jazyk obsahovat mnoho klíčových slov

¹⁸ Unicode je kódování textu tak, aby umožňoval zobrazení jazyka jakéhokoli státu, např. českého jazyka, pro zakódování jednoho znaku této sady je zapotřebí víc než jeden byte, jak tomu je u standardní znakové sady ANSI

a knihoven specifických pro technologie a produkty společnosti Microsoft. Výsledkem byl naprostý chaos.[5]

Nyní vstupujeme do prostředí .NET, zcela nového pracovního prostředí, jež musí obsahovat nová rozšíření obou popisovaných jazyků. Společnost Microsoft obešla tento problém tím, že do jazyku C++ přidala další specifická klíčová slova a jazyk Visual Basic přetavila na zcela nový jazyk VB.NET. Ten sice má určité prvky původní syntaxe jazyka Visual Basic, ale je tak jiný, že jej z praktického hlediska můžeme považovat za úplně nový programovací jazyk.

V tomto kontextu se společnost Microsoft rozhodla poskytnout vývojářům další možnosti, což je jazyk navržený speciálně pro platformu .NET a vytvořený úplně na nových základech. Výsledkem je jazyk C#. Microsoft oficiálně popisuje tento jazyk jako „jednoduchý, moderní, objektově orientovaný a typově spolehlivý programovací jazyk, odvozený od jazyků C a C++“. Většina nezávislých pozorovatelů by pravděpodobně konec této definice změnila na „odvozený od jazyků C, C++ a Java“. Tyto popisy jsou technicky přesné, ale o kráse jazyka říkají málo. Syntakticky je jazyk C# velmi podobný jak jazyku C++, tak jazyku Java a to do té míry, že většina klíčových slov je naprosto stejných. Kromě toho jazyk C# používá stejnou strukturu bloku ohraničeného složenými závorkami ({ }). Jednotlivé příkazy jsou odděleny středníky. První dojem z kódu jazyka C# je, že se díváte na kód v jazyku C++ nebo Java. Jeho návrh je však s moderními vývojářskými nástroji sladěn mnohem více než v případě obou zmiňovaných protějšků. Jazyk C# byl navržen tak, aby poskytoval jednoduchost práce v jazyce Visual Basic a obrovský výkon a nízkoúrovňový přístup k paměti jazyka C++. Mezi klíčové vlastnosti jazyka C# patří:

- Plná podpora tříd a objektově orientovaného programování, včetně dědění rozhraní a implementace, virtuálních funkcí a přetěžování operátorů a funkcí
- Konzistentní a dobře definovaná množina základních typů
- Vestavěná podpora automatického generování dokumentace ve formátu XML
- Automatické čištění přidělované paměti
- Možnost označení tříd nebo metod uživatelsky definovanými atributy. To může být užitečné při tvorbě dokumentace a může mít určitý vliv na překlad (například označení metody tak, aby byly přeloženy pouze v ladících sestaveních aplikace)

- Úplný přístup k základní knihovně tříd prostředí .NET a stejně snadný přístup k rozhraní Windows API (pokud jej budeme potřebovat)
- Ukazatele a bezprostřední přístup k paměti na vyžádání. Jazyk byl však navržen tak, abychom s těmito prostředky prakticky vůbec nemuseli pracovat
- Podpora vlastností a událostí jako v jazyce Visual Basic
- Možnost překladu kódu do aplikace nebo knihovny komponent prostředí .NET (které lze volat stejným způsobem jako ovládací prvky **ActiveX**¹⁹ nebo komponenty modelu COM) pouhou změnou volby nastavení překladače.
- Možnost užití jazyka C# pro tvorbu dynamických webových stránek, založených na technologii ASP.NET.

I když je jazyk C# v mnohém podobný jazyku Java, má určitá vylepšení, konkrétně jazyk Java není navržen k tomu, aby byl spouštěn v pracovním prostředí .NET.[5]

Pro úplnost bych se měl pozastavit u několika omezení jazyka C#. Patří k nim především skutečnost, že tento jazyk není navržen pro úlohy spouštěné v režimu reálného času nebo pro tvorbu extrémně výkonného kódu - takového, v němž se musíme strachovat, zda cyklus bude vyžadovat 5000 nebo 5100 cyklů procesoru a v němž musíme uvolnit nepotřebné prostředky v mikrosekundách. V této oblasti bude patrně i nadále dominovat jazyk C++. Jazyk C# totiž postrádá určité schopnosti pro extrémně výkonné aplikace, mezi ně patří schopnost určovat vložené funkce a destruktory, u nichž je zaručeno, že při zpracování kódu proběhnou v reálném čase. Množství aplikací, které do této kategorie spadají, je však velmi málo.

¹⁹ ActiveX prvky jsou objekty typu COM vyvinuté společností Microsoft pro rychlejší vývoj aplikací pro operační systém Windows

2.5 Závislost jazyka C# na prostředí .NET

C# je nový programovací jazyk významný především ze dvou hlavních důvodů:

- I. Je navržen speciálně pro použití v pracovním prostředí .NET Framework (pro platformu bohatě vybavenou pro vývoj, šíření a spouštění distribuovaných aplikací)
- II. Je to jazyk založený na moderní technologii objektově orientovaného návrhu. Při jeho návrhu se mohli programátoři ve společnosti Microsoft opřít o více než dvacetileté zkušenosti z vývoje podobných jazyků, tedy z celé etapy vývoje objektově orientovaných technologií.

Je třeba si uvědomit, že jazyk C# má své vlastní pravidla. Přestože je navržen proto, aby generoval kód pro prostředí .NET, není jeho součástí. Existují určité funkce, jež jsou dostupné v prostředí .NET, ale ne v jazyce C#. Stejně tak nás může překvapit zjištění, že jazyk C# obsahuje funkce a vlastnosti, které naopak v prostředí .NET nenajdete (např. přetěžování operátorů).

Nesmíme však zapomenout na skutečnost, že jazyk C# je určen pro použití v prostředí .NET. Proto musíme pochopit, v jakém prostředí pracuje. Jedině tak jej budeme schopni využívat efektivně. Jazyk C# tedy generuje aplikace, které je možné spustit pouze za předpokladu, že máme na počítači s operačním systémem Windows nainstalované prostředí .NET Framework. Toto prostředí poskytuje společnost Microsoft zcela zdarma.[4]

2.6 Společné prostředí zpracování

Centrem pracovního prostředí .NET Framework je jeho operační prostředí (nebo také prostředí zpracování) označované jako modul **CLR** (Common Language Runtime),

nebo operační prostředí .NET. Kód spouštěný pod kontrolou operačního prostředí .NET se často označuje pojmem *spravovaný kód* (*Managed Code*²⁰).

Předtím, než kód v prostředí .NET (CLR) spustíme, je třeba kód (vyvinutý v jazyce C# nebo v jiném jazyce) přeložit. Překlad (kompilace) programů v prostředí .NET má dva stupně:

- I. Nejprve je kód přeložen do jazyka **MSIL**, zkráceně jen **IL**
- II. Následně je kód IL přeložen modulem CLR na kód specifický pro danou platformu

Na první pohled je kompilační proces zdlouhavý. Jenže rozdělení procesu na dva stupně je důležité. Klíčem k mnoha výhodám prostředí .NET je totiž existence jazyka IL (spravovaného kódu). IL je mezistupeň při kompilaci, který si lze představit, jako nespustitelný kód, do kterého se převádí zdrojový kód vytvořený programátorem v jakémkoli programovacím jazyku, podporovaném prostředím .NET. To má obrovskou výhodu v tom, že lze psát různé části aplikace (projektu) v různých programovacích jazycích podporovaných prostředím .NET a kompilaci budou tyto součásti aplikace (různé dynamické knihovny DLL) dokonale spolupracovat jako by byly napsané ve stejném programovacím jazyce.[6]

2.7 Výhody překladu do jazyka IL

Jazyk IL sdílí s bajtovým kódem jazyka Java myšlenku, že do nativního kódu cílového počítače může být velmi rychle překládán nízkourovňový jazyk s jednoduchou syntaxí (založený na číselných kódech, nikoli na textu). Existence dobře definované univerzální syntaxe kódu přináší významné výhody:

- **Nezávislost na platformě.** To znamená, že stejný soubor obsahující instrukce bajtového kódu lze umístit na jakoukoli platformu

²⁰ Robinson, S., C# Programujeme profesionálně. Brno: Computer Press, 2003. ISBN 80-251-0085-5

- **Vylepšení výkonu.** Jazyk IL překládá kód jen v případě potřeby na rozdíl od jazyka Java, tomuto procesu se říká **JIT** (*Just-In-Time*).
- **Interoperabilita jazyků.** Zjednodušeně lze říci, že je možné do jazyka IL překládat zdrojové kódy z jednoho jazyka a výsledek bude interoperabilní²¹ s kódem, který byl do jazyka IL přeložen z jiného programovacího jazyka.[6]

V přílohách je uveden příklad zápisu jazyka C# a MSIL.

²¹ Interoperabilita znamená „navzájem kompatibilní“ nebo také „vzájemná kompatibilita“

II. PRAKTICKÁ ČÁST

3 ZÁKLADNÍ KONCEPCE APLIKACE

3.1 Popis použitých ovládacích prvků

V horní části aplikace *SQLCommander* je „roletkové“ menu a nástrojová lišta. V menu lze nalézt všechny základní funkce aplikace a nástrojová lišta obsahuje nejpoužívanější nástroje aplikace. Podrobný popis menu i nástrojové lišty je popsán níže.

3.1.1 Hlavní menu

V případě aplikace *SQLCommander* jsem volil vytvoření statického menu, které se mi jevilo pro daný typ aplikace jako nejvýhodnější. Statické menu znamená, že položky v menu nejsou dynamicky generovány za běhu aplikace, ale jsou vytvořeny při startu aplikace a za chodu aplikace se již nemění. Aplikace *SQLCommander* poskytuje relativně velké množství funkcí a nebylo by možné tyto funkce seskupit pouze do ovládacích prvků hlavního formuláře aplikace resp. do panelů aplikace. Na druhou stranu nebylo zapotřebí volit možnost generovat položky hlavního menu dynamicky za běhu aplikace, protože každá položka menu má zhruba do deseti položek a je dostatečně přehledná i když zobrazuje i položky, které by v danou chvíli mohli být skryté. Položky, které jsou v daný okamžik nevyužívané, jsou zakázané a mají světle šedou barvu.

3.1.2 Nástrojová lišta

Nástrojová lišta slouží pro rychlý přístup k funkcím aplikace, bez složitého hledání dané funkce v hlavním menu aplikace, obsahuje nejčastěji používané funkce. Nástrojová lišta neboli **ToolBar** má často určitou skupinu podobných funkcí, reprezentovanou tlačítky s *ikonami*²². Pro tuto aplikaci jsem použil „plovoucí nástrojové lišty“ (Floating ToolBars),

²² Ikona je ovládací prvek aplikace, reprezentovaný malým obrázkem, který svým vzhledem specifikuje funkci daného ovládacího prvku. Např. tlačítka na nástrojové liště

kteřé mají tu výhodu, že je možné měnit jejich pozici stisknutím myši a tažením na žádané místo v aplikaci, po okraji hlavního okna aplikace. Tím lze dosáhnout uživatelsky přívětivější rozhraní, které nabízí uživateli definovat si pozici nástrojových panelů dle vlastního uvážení. Samozřejmě, že pozice kam lze panel přesunout je omezená a to z důvodu zajištění neměnné pozice jiných ovládacích prvků. Poslední funkcí, která umožňuje zvýšit flexibilitu aplikace je možnost skrýt nebo zobrazit tyto nástrojové lišty.

3.1.3 Rozložení panelů

Při návrhu jakékoli aplikace je velmi důležité po stránce uživatelské přívětivosti a efektivity práce, dobré rozložení panelů aplikace. Pro tento druh aplikace jsem použil tři panely, u kterých je možné měnit jejich velikost a tím pádem, je možné obsah jednotlivých panelů přizpůsobovat jejich aktuálnímu obsahu.[8]

Další důležitou vlastností panelů aplikace je možnost jejich skrývání. Skrývání panelů využívá především zkušený uživatel, který v danou chvíli pracuje pouze s omezeným množstvím panelů a ostatní skryje. Skrytím nepotřebných panelů získá uživatel více prostoru pro obsah viditelných panelů. Pokud si uživatel nastaví optimální rozložení panelů, jeho práce se zefektivní a urychlí, čímž ho práce s aplikací stojí méně času a tím pádem i peněz. I na toto kritérium byl kladen důraz při výběru správného typu a rozložení panelů v aplikaci *SQLCommander*.

3.1.4 SQLCommander je vícevláknová aplikace

Většina jednoduchých aplikací běží pouze v jednom vláknu tzn., má pouze jeden vstupní bod a to metodu, která se ve většině programovacích jazyků nazývá *main()*. Běh takovéto aplikace začíná zpracováním prvního příkazu v této metodě a končí ukončením metody.[5]

Taková struktura je dostačující pro všechny programy, v nichž existuje určitá posloupnost úloh. Nevýhodou tohoto řešení je, že aplikace, která využívá většinu času např. na matematické výpočty, se při zpracování těchto výpočtů nedá využívat pro další zpracování jiných výpočtů a je zapotřebí počkat na dokončení již spuštěného procesu a poté je možné spustit další proces. Pro aplikaci *SQLCommander* je však řešení s jedním

vstupním bodem (*Singlethreaded*) nedostačující a proto jsem zvolil vícevláknový typ aplikace (*Multithreaded*).

V programech se často vyskytují situace, v nichž je třeba v jednom okamžiku vykonávat více úloh. Aplikace *SQLCommander* funguje na velmi podobném principu, jako např. aplikace *Internet Explorer*²³, která pro každou záložku prohlížeče vytváří nové vlákno, které nezávisle na ostatních vláknech zpracovává požadavky od uživatele a tím pádem nemusí uživatel čekat na ukončení načtení dat jiné záložky.

Vlákno jak jsem již podotkl, je posloupností instrukcí zpracovaných počítačem. Je opravdu mnoho důvodů, proč by aplikace měla být vícevláknová. Počet vláken, které lze spustit z aplikace je „neomezený počet“²⁴. Vývojáři stačí pouze určit metodu, která nové vlákno zahájí. První vlákno v aplikaci začíná vždy v metodě `main()`, protože aplikaci spouští operační prostředí platformy .NET Framework (.NET Runtime) a to hledá právě metodu `main()` jako začátek aplikace. Další vlákna již spouští naše aplikace, což znamená, že je na ní, kolik dalších vláken spustí.[8]

3.1.5 Zpracování více vláken v aplikaci *SQLCommander*

Doposud jsem o vícevláknové aplikaci hovořil spíše obecně, ale z mého popisu vyplývá, že vlákna v aplikaci *SQLCommander* běží simultánně²⁵. Ve skutečnosti však jeden procesor může v jednom okamžiku vykonávat pouze jednu činnost. Máme-li **víceprocesorový**²⁶ systém, pak je teoreticky možné, že v jednom okamžiku jsou v systému vykonávány dvě a více operací - každá v jednom procesoru. Většina systémů je však **jednoprocesorových**, takže jak mohou činnosti probíhat simultánně? Systém Windows

²³ Internet Explorer je aplikace společnosti Microsoft určená pro prohlížení webových stránek

²⁴ Počet vláken které mohou být spuštěny v metodě `main()` je omezený operačním systémem a programovacím jazykem, resp. Množstvím volných systémových prostředků OS (paměť, výkon procesoru)

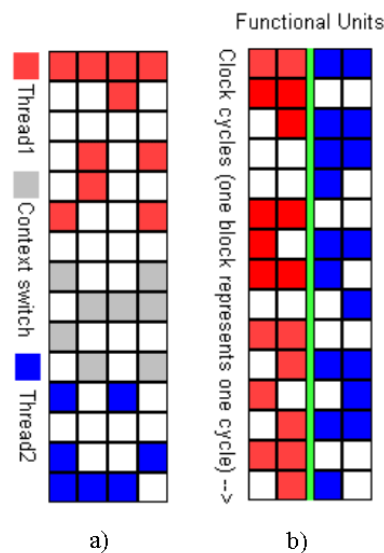
²⁵ simultánně znamená paralelně s něčím dalším, nezávisle na něčem dalším

²⁶ Víceprocesorový znamená počítač (platformu) s více procesory

vytváří zdání, že v jednom okamžiku probíhá více operací. Zmiňovaný mechanismus je označován jako **víceúlohové zpracování**.

Co si pod tímto pojmem představit? Systém Windows vezme vlákno v určitém procesu a nechá je velice krátkou dobu běžet. Dokumentace společnosti Microsoft neuvádí, jak dlouhá tato perioda je. Jde totiž o jeden z interních parametrů operačního systému, které systém Windows nastavuje pro dosažení optimálního výkonu. V každém případě to není informace, bez níž bychom nebyli schopni aplikaci v systému Windows spustit. V našem chápání jde každopádně o velmi krátkou dobu, zcela jistě nepřekračující milisekundy. Tato perioda se označuje za **dávku času** (time slice). Po jejím ukončení přebírá systém Windows kontrolu zpět a přiděluje další dávku jinému vláknu. Vzhledem k tomu, že jsou zmiňované dávky času velmi krátké, získáme dojem, jakoby se věci v počítači odehrávali simultánně.

Z popisu uvedeného výše je zřejmé, že volba vícevláknové aplikace má opravdu mnoho výhod a proto jsem aplikaci *SQLCommander* navrhl také jako vícevláknovou. Obrázek 2 vysvětluje proces spouštění dvou vláken simultánně na jednoprocessorové a víceprocessorové platformě.



Obrázek 2: Zpracování dvou vláken a) jeden procesor b) dva procesory²⁷

²⁷ Zdroj [<http://www.slcentral.com/articles/01/6/multithreading/print.php>]

3.1.6 TreeView

Pro zobrazení seznamu připojení a jejich položek byl použit ovládací prvek s názvem *TreeView*. Tento ovládací prvek zobrazuje položky jako „rozbalovací“ strom, kde každá položka je zobrazena v jednom řádku a na jeho podsložky se dostaneme pomocí ikony „+“ umístěné v levé části zvolené položky.

Tento ovládací prvek má pro zobrazování dat jako „zabalených“ položek tu výhodu, že pokud chceme pracovat pouze se zvolenými položkami, tak si ostatní skryjeme, nebo zobrazíme stisknutím ikony „+“ v levé části položky a tím se nám uvolní prostor na zobrazení dalších položek. Uživatel nemusí používat posuvník okna k tomu, aby se dostal na ostatní položky, které jsou umístěny níže v panelu s aktuálními připojenými RDBMS.

3.1.7 TabControl

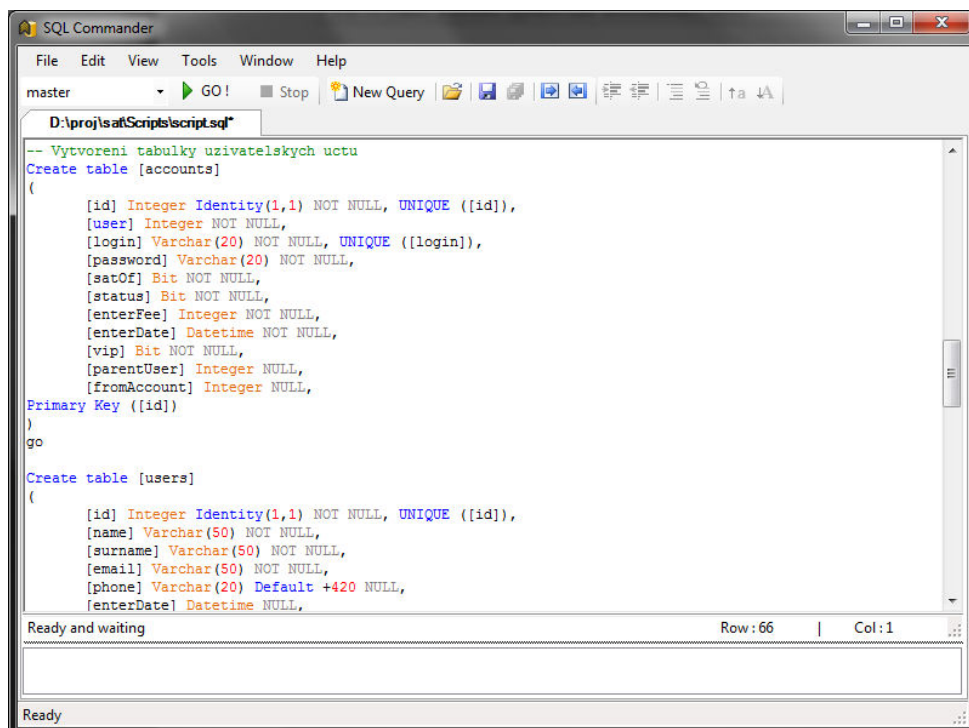
TabControl je uživateli dobře známý grafický ovládací prvek, který zobrazuje další obsah na panelech, které lze přidávat formou dalších záložek do tohoto ovládacího prvku. V panelech záložek jsou další ovládací prvky, které je možné ovládat po přepnutí na tento panel pomocí záložky v horní nebo spodní části ovládacího prvku *TabControl*. Toto řešení je velmi často používáno při zobrazování panelů s různým obsahem, kdy je vidět pouze panel, se kterým právě pracujeme zaplňující většinu místa a tím pádem ovládací prvky na tomto panelu mají dostatečnou velikost pro pohodlnou práci s nimi. *TabControl* byl použit v aplikaci *SQLCommander* pro zobrazování panelů, kde se v horní části interpretují SQL dotazy na připojený RDBMS a ve spodní části je po vykonání SQL dotazu zobrazen výsledek.

3.1.8 SQLSyntaxTextBox

Tento ovládací prvek umožňuje uživateli vkládat a editovat text dotazů jazyka SQL v aplikaci *SQLCommander*. *SQLSyntaxTextBox* také poskytuje mnoho dalších formátovacích prvků, které standardní ovládací prvek *TextBox*, ze kterého je *SQLSyntaxTextBox* odvozen, neumožňuje. Text může být vkládán přímo psaním

z klávesnice, může být vložen otevřením nového souboru a zároveň tento text můžeme uložit do souboru s námi zvoleným názvem. Hlavní výhodou, která je využita v aplikaci *SQLCommander* je ta vlastnost, že je možné barevně zvýrazňovat editovaný text a tím rozlišit elementy SQL dotazů. Této vlastnosti se také říká „zvýrazňování syntaxe“ nebo také zvýrazňování klíčových slov od běžného textu. V aplikaci *SQLCommander* jsem naprogramoval ovládací prvek *SQLSyntaxTextBox* takovým způsobem, že dokáže zvýrazňovat nejenom syntaxi klíčových slov jazyka SQL, ale také rozlišuje dva typy klíčových slov, datové typy jazyka SQL, řetězce, čísla a komentáře. Při psaní SQL dotazů se při každé změně textu vyvolá událost **OnTextChanged** a prověří se všechny elementy editovaného textu a pokud systém rozpozná, že tento element je některé ze zvýrazňovaných objektů jazyka SQL,[14] zvýrazní je definovanou barvou, kterou je možné definovat v nastavení aplikace *SQLCommander*.

SQLSyntaxTextBox v aplikaci *SQLCommander* pomáhá uživateli k lepší orientaci v psaní SQL dotazů barevným zvýrazňováním textu, čímž se stávají SQL dotazy přehlednějšími. Na obrázku Obrázek 3 je příklad zvýrazňování syntaxe jazyka SQL, aplikací *SQLCommander*.



The screenshot shows the SQL Commander application window. The title bar reads "SQL Commander". The menu bar includes "File", "Edit", "View", "Tools", "Window", and "Help". The toolbar contains icons for "GO!", "Stop", "New Query", and other standard database operations. The main text area displays the following SQL code with syntax highlighting:

```
-- Vytvoreni tabulky uzivatelskych uctu
Create table [accounts]
(
    [id] Integer Identity(1,1) NOT NULL, UNIQUE ([id]),
    [user] Integer NOT NULL,
    [login] Varchar(20) NOT NULL, UNIQUE ([login]),
    [password] Varchar(20) NOT NULL,
    [satOf] Bit NOT NULL,
    [status] Bit NOT NULL,
    [enterFee] Integer NOT NULL,
    [enterDate] Datetime NOT NULL,
    [vip] Bit NOT NULL,
    [parentUser] Integer NULL,
    [fromAccount] Integer NULL,
    Primary Key ([id])
)
go

Create table [users]
(
    [id] Integer Identity(1,1) NOT NULL, UNIQUE ([id]),
    [name] Varchar(50) NOT NULL,
    [surname] Varchar(50) NOT NULL,
    [email] Varchar(50) NOT NULL,
    [phone] Varchar(20) Default +420 NULL,
    [enterDate] Datetime NULL,
```

 The status bar at the bottom indicates "Ready and waiting" and "Row: 66 | Col: 1".

Obrázek 3: Zvýrazňování syntaxe jazyka SQL. Zdroj [Vlastní zpracování]

3.1.9 DataGridView

Poslední důležitý ovládací prvek aplikace *SQLCommander* je *DataGridView*. Tento ovládací prvek slouží pro zobrazení dat v tabulkové podobě a tyto data lze přehledně prohlížet nebo editovat. První řádek zobrazuje názvy sloupců dat. Při editaci dat v *DataGridView* je v prvním sloupci zobrazen příznak editace, což je ikona s psacím perem. Pokud *DataGridView* v editačním módu zobrazuje data z tabulky, která má primární klíče, je tento sloupec podbarven odlišnou barvou pro lepší orientaci v typech sloupců. Sloupce v *DataGridView* lze také pro lepší orientaci v tabulce řadit jak vzestupně, tak sestupně a u všech sloupců zobrazené tabulky v *DataGridView* lze měnit jejich velikost.

3.2 Popis ovládání aplikace SQLCommander

3.2.1 Připojení k RDBMS

Při spuštění aplikace *SQLCommander* se uživateli zobrazí *SplashScreen*²⁸ a po jeho zmizení, formulář pro připojení k RDBMS. V této verzi aplikace *SQLCommander* jsou podporovány databázové systémy MsSQL, MySQL a Firebird (Firebird není dostatečně otestován). Formulář pro připojení k RDBMS má tři hlavní části, ve kterých je možné definovat vlastnosti připojení. Všechny tři části jsou zobrazeny v záložkách ovládacího prvku *TabControl*. Spodní část připojovacího formuláře tvoří tři tlačítka.

Popis funkce tlačítek ve spodní části připojovacího formuláře:

- **Connect**, slouží pro odeslání zadaných nebo změněných údajů připojení aplikaci, která se s těmito údaji pokusí spojit s databázovým systémem.
- **Close(Cancel)**, zavře formulář pro připojení k databázovému systému, nebo pokud byl odeslán požadavek na připojení k databázovému systému, tento požadavek zruší a je možné se opět připojovat k jinému databázovému systému.

²⁸ *SplashScreen* je úvodní obrázek aplikace po spuštění, který po několika vteřinách zmizí

- **Help**, funkce tohoto tlačítka je v současné verzi aplikace potlačena. V budoucnu bude toto tlačítko sloužit k zobrazení nápovědy aplikace *SQLCommander*.

3.2.1.1 Záložka Login

Je to nejdůležitější část přihlašovacího formuláře. Slouží pro zadávání nebo změnu nezbytných údajů připojení.

Popis jednotlivých ovládacích prvků:

- **Provider**, zde se volí poskytovatel databázového systému. V současné verzi aplikace je možné zvolit ze tří poskytovatelů (výrobců RDBMS) databázového systému - MS SQL, MySQL, Firebird (pouze testovací verze).
- **Server name**, je doménové jméno počítače (DNS), nebo IP-Adresa²⁹ počítače databázového systému, ke kterému se budeme připojovat.
- **Authentication**, slouží pro volbu *Windows Authentication* nebo *SQL Server Authentication*. Toto nastavení se používá pouze při připojování k MS SQL databázovému systému, u ostatních se tato volba nevyužívá.
- **User name**, zde vložíme uživatelské jméno, kterým se budeme přihlašovat k danému databázovému systému (při volbě *Windows Authentication* se tento ovládací prvek nevyužívá)
- **Password**, zde vložíme heslo k příslušnému uživatelskému jménu (při volbě *Windows Authentication* se tento ovládací prvek nevyužívá)
- **Remember password**, tento ovládací prvek se v aktuální verzi nevyužívá, z důvodu vyšší bezpečnosti uživatelských údajů (v další verzi bude využito pro zapamatování vložených uživatelů a jejich hesel).

²⁹ IPA Internet Protocol Address, je adresa počítače v síti, používaná protokolem IP pro jasnou definici počítače při komunikaci, IPv4 se skládá se ze čtyř číslic v rozsahu 0 až 255, IPv6 se skládá se ze šesti číslic v rozsahu 0 až 255

3.2.1.2 Záložka *Connection Specification*

V této záložce jsou doplňující informace, které je možné specifikovat u připojení k RDBMS.

Popis jednotlivých ovládacích prvků:

- **Connect to database**, v tomto ovládacím prvku je možné si zvolit databázi, ke které budeme při přihlášení k databázovému systému připojeni.
- **Network packet size [bajty]**, slouží pro nastavení velikosti přenášených paketů síťové komunikace mezi klientským PC (osobním počítačem) a serverem SQL.
- **Connection timeout [sekundy]**, je nastavení doby po jakou se bude snažit aplikace *SQLCommander* připojovat ke specifikovanému databázovému systému
- **Minimum pool size**, slouží pro zadání minimálního množství udržovaných připojení, viz MSDN
- **Maximum pool size**, slouží pro zadání maximálního množství udržovaných připojení, viz MSDN
- **Pooling**, toto zaškrtačkové políčko aktivuje nebo deaktivuje *Pooling*³⁰ tohoto připojení
- **Replication**, při aktivaci této volby bude SQL server využíván i po dobu jeho replikace³¹ s jinými SQL servery.
- **Encrypt connection**, slouží pro využívání šifrované komunikace mezi klientským PC a SQL serverem
- **Reset All**, toto tlačítko nastaví všechny hodnoty ovládacích prvků této záložky na jejich výchozí hodnoty.

³⁰ Kompletní specifikace systému Pooling je dostupná na této adrese [http://msdn.microsoft.com/en-us/library/8xx3tyca\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/8xx3tyca(VS.80).aspx)

³¹ Replikace je odevzdání nových dat jiným SQL serverům a uložení nových od jiných SQL serverů, je to forma zálohování dat mezi SQL servery

3.2.1.3 Záložka *Expert Settings*

V poslední záložce přípojovacího formuláře je možné detailně nastavovat parametry daného SQL připojení. Hodnoty v této záložce se doporučuje měnit pouze zkušeným uživatelem, protože mají velký vliv na chování nastavovaného SQL připojení a všechny tyto parametry je zapotřebí dobře znát.

Hlavní část této záložky tvoří ovládací prvek *DataGridView*, který se využívá při zobrazení vlastností určitého objektu a hodnot těchto vlastností. V horní části *DataGridView* je možné měnit řazení zobrazených vlastností do skupin, nebo abecedně. Pokud je seznam vlastností připojení seřazen dle kategorií, lze tyto kategorie zobrazovat nebo skrývat znakem „+“ u příslušné kategorie objektu *DataGridView*. V levé části *DataGridView* je název vlastnosti, v pravé části je k této vlastnosti její hodnota, kterou lze měnit. Ve spodní části je tlačítko **Reset All**, které nastaví hodnoty všech vlastností zobrazených v *DataGridView* na jejich výchozí hodnoty.

3.2.2 Menu a nástrojové lišty

3.2.2.1 Hlavní menu aplikace

Lze v něm najít všechny funkce, které aplikace *SQLCommander* nabízí. Menu se nachází v horní části aplikace a všechny jeho funkce jsou rozděleny přehledně do sekcí (roletek), kde každá sekce obsahuje funkce jednoho typu. Hlavní menu aplikace *SQLCommander* je standardní objekt, se kterým se setkáme i v ostatních aplikacích pro operační systém Windows.

Popis jednotlivých položek menu:

- **File** - Tato sekce obsahuje základní funkce aplikace, viz Tabulka 4.

Tabulka 4: Popis záložky *File* v hlavním menu. Zdroj [Vlastní zpracování]

Název položky	Popis položky	Kláv. Zkr.
Connect to SQL	Otevře formulář nového připojení	Není
Disconnect from	Odpojí vybrané připojení SQL serveru	Není

SQL		
New Query	Vytvoří novou záložku pro formulování SQL dotazu	Ctrl + N
Open Query	Otevře soubor v počítači s SQL dotazy do nové záložky	Ctrl + O
Close	Zavře vybranou záložku	Ctrl + F4
Save	Uloží SQL dotazy z právě vybrané záložky do souboru	Ctrl + S
Save As	Uloží SQL dotazy z právě vybrané záložky do souboru s názvem souboru zvoleným uživatelem	Není
Save All	Uloží všechny neuložené SQL dotazy v záložkách	Ctrl + Shift + S
Page Setup	V této verzi aplikace NEVYUŽITO	Není
Print	V této verzi aplikace NEVYUŽITO	Ctrl + P
Recent Files	V této verzi aplikace NEVYUŽITO	Není
Exit	Ukončí aplikaci SQLCommander	Alt + X

- **Edit** – Tato sekce obsahuje funkce pro editaci textu SQL dotazů, viz Tabulka 5.

Tabulka 5: Popis záložky *Edit* v hlavním menu. Zdroj [Vlastní zpracování]

Název položky	Popis položky	Kláv. Zkr.
Undo	Vrátí zpět předešlou změnu v editaci textu SQL dotazu	Ctrl + Z
Redo	Provede opět vrácenou změnu v editaci textu SQL dotazu	Ctrl + Y
Cut	Vyřízne vybraný text do schránky	Ctrl + X
Copy	Zkopíruje vybraný text do schránky	Ctrl + C

Paste	Vloží text ze schránky na pozici kurzoru klávesnice	Ctrl + V
Delete	Smaže označený text SQL dotazu	Shift + Del
Select All	Označí všechny text SQL dotazu ve zvolené záložce	Ctrl + A
Find	V této verzi aplikace NEVYUŽITO	Ctrl + F
Replace	V této verzi aplikace NEVYUŽITO	Ctrl + H

- **View** - Tato sekce obsahuje funkce pro zobrazování nebo skrývání panelů a nástrojových lišt, viz Tabulka 6.

Tabulka 6: Popis záložky *View* v hlavním menu. Zdroj [Vlastní zpracování]

Název položky	Popis položky	Kláv. Zkr.
Panels	Umožňuje nastavovat viditelnost levého a pravého panelu	F7, F8
Toolbars	Umožňuje nastavovat viditelnost nástrojových lišt	Není
Refresh	Aktualizuje vybranou část <i>TreeView</i> v levém panelu	Ctrl + R

- **Tools** - Tato sekce obsahuje funkce pro nastavení aplikace, speciální funkce pro práci s textem a vykonávání dotazů, viz Tabulka 7.

Tabulka 7: Popis záložky *Tools* v hlavním menu. Zdroj [Vlastní zpracování]

Název položky	Popis položky	Kláv. Zkr.
Customize	V této verzi aplikace NEVYUŽITO	Není
Options	Otevře formulář s nastavením aplikace	Není

Text Gadgets	V této verzi aplikace NEVYUŽITO	Není
Run Query	Spustí SQL dotazy ve zvolené záložce	F5

- **Window** - Tato sekce obsahuje funkce pro práci se záložkami, viz Tabulka 8.

Tabulka 8: Popis záložky *Window* v hlavním menu. Zdroj [Vlastní zpracování]

Název položky	Popis položky	Kláv. Zkr.
Next Pane	Zvolí následující záložku vpravo od vybrané	Není
Previous Pane	Zvolí předešlou záložku vlevo od vybrané	Není
Close All Panes	Zavře všechny otevřené záložky (pokud některá ze záložek má neuložený text SQL dotazu, zeptá se aplikace na uložení textu)	Není
Panes	V této verzi aplikace NEVYUŽITO	Není

- **Help** - Tato sekce obsahuje nápovědu aplikace, viz Tabulka 9.

Tabulka 9: Popis záložky *Help* v hlavním menu. Zdroj [Vlastní zpracování]

Název položky	Popis položky	Kláv. Zkr.
Index	V této verzi aplikace NEVYUŽITO	F1
About	Otevře formulář s informacemi o aplikaci SQLCommander	Není

3.2.2.2 *Nástrojové lišty aplikace – Toolbars*

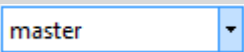


Hlavní menu nám poskytuje kompletní seznam funkcí aplikace, rozdělených do sekcí. Avšak pracovat s funkcemi v hlavním menu by bylo velmi často pomalé a zdlouhavé. Pro tyto případy jsou v aplikacích používány nástrojové lišty nebo také *Toolbars*, které také slučují funkce jednoho typu do jednoho panelu s tím rozdílem, že panely je možné mít stále zobrazeny, tím pádem lze daleko rychleji využívat jejich funkce. Každá funkce na panelu je

reprezentována ikonou, která svým obrázkem jednoznačně vysvětluje její funkci. Po kliknutí myši na zvolenou ikonu se aktivuje funkce, kterou reprezentuje.

Aplikace *SQLCommander* se skládá ze tří panelů, které jsou popsány níže.

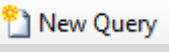





- **DB editor** - je panel pro práci změnu aktuální databáze a spouštění či zastavování SQL dotazu

Tabulka 10: Popis tlačítek nástrojové lišty DB Editor. Zdroj [Vlastní zpracování]

Tlačítko nástr. lišty	Popis funkce
	Slouží pro změnu databáze, ve které se budou vykonávat spouštěné SQL dotazy ze zvolené záložky
	Slouží pro spuštění všech SQL dotazů ve zvolené záložce (pokud je označený text SQL dotazu, spustí se pouze vybraný SQL dotaz)
	Zastaví právě vykonávaný SQL dotaz ve zvolené záložce

- **Standard** je panel poskytující tlačítka pro práci se záložkami editoru a práci se soubory SQL dotazů

Tabulka 11: Popis tlačítek nástrojové lišty *Standard*. Zdroj [Vlastní zpracování]

Tlačítko nástr. lišty	Popis funkce
	Vytvoří novou záložku pro editaci textu SQL dotazu
	Otevře soubor s textem SQL dotazu a zobrazí jej v nově vytvořené záložce
	Uloží text SQL dotazu právě zvolené záložky do souboru
	Uloží text SQL dotazů ve všech záložkách aplikace
	Skryje levý panel s připojenýma RDBMS
	Skryje pravý panel s vlastnostmi RDBMS

- **Text editor** - je panel poskytující funkce pro práci s textem. V této verzi aplikace je tento panel nevyužitý. Jeho funkčnost bude dopracována v některé další verzi.

3.2.3 Levý panel aplikace

V levém panelu aplikace *SQLCommander* jsou zobrazeny všechny připojení k databázovým systémům a jejich obsah je zobrazen ve stromové struktuře ovládacího prvku *TreeView*.

První úroveň stromové struktury (počítáno od jedné) obsahuje vždy název připojení databázového systému, tzn. IP adresu nebo DNS jméno počítače s SQL službou, popř. název instance databázového systému (pokud má daný databázový systém více instancí SQL služby).

Kontextové menu obsahuje tyto funkce:

- **Set as Working Server** - Nastaví toto připojení jako aktuální, na které se budou provádět spouštěné SQL dotazy
- **Connect to Server** - Otevře formulář pro nové připojení k SQL serveru
- **Disconnect from Server** - Odpojí toto SQL připojení
- **New Query** - Otevře novou záložku pro psaní SQL dotazů

Druhá úroveň stromové struktury obsahuje položku *Databases*, která pod sebou obsahuje seznam dostupných databází tohoto SQL připojení a dále obsahuje položku *Logins*, která pod sebou obsahuje seznam všech uživatelských účtů tohoto SQL připojení. Jednotlivé databáze dále zobrazují seznam tabulek. Poklepáním na název tabulky se zobrazí seznam sloupců této tabulky a jejich typ, včetně doplňkových informací o každém sloupci. Na konci druhé úrovně je zobrazen v kulatých závorkách počet elementů této položky tzn. u položky *Databases* je zobrazen počet databází, u položky *Logins* je zobrazen počet uživatelských účtů a u názvů tabulek je zobrazen počet záznamů v tabulce. Zobrazení počtu elementů ve stromové struktuře závisí na nastavení aplikace ve formuláři *Settings*, kterou lze vyvolat z hlavního menu aplikace.

Kontextové menu *Databases* a *Logins* obsahuje tyto funkce:

- **Refresh** - provede obnovení dané položky

Kontextové menu každé databáze obsahuje tyto funkce:

- **New Query** - Otevře novou záložku pro psaní SQL dotazů
- **Rename** - Přejmenuje název databáze (platí pouze pro MSSQL)
- **Delete** - Smaže databázi
- **Refresh** - Obnoví obsah této položky




Kontextové menu každé tabulky obsahuje tyto funkce:

- **Open table** - Otevře novou záložku se zobrazením obsahu této tabulky
- **Rename** - Přejmenuje název tabulky
- **Delete** - Smaže tabulku

3.2.3.1 Nástrojová lišta levého panelu

V horní části levého panelu je nástrojová lišta, popis jednotlivých tlačítek je v Tabulce 12.

Tabulka 12: Popis tlačítek nástrojové lišty levého panelu. Zdroj [Vlastní zpracování]

Tlačítko nástr. lišty	Popis funkce
	Otevře formulář pro nové připojení SQL serveru
	Odpojí toto SQL připojení
	Obnoví obsah prvku <i>TreeView</i> pro zobrazení SQL připojení

3.2.4 Střední panel aplikace

Ve středním panelu jsou nejdůležitější ovládací prvky aplikace *SQLCommander*, které tvoří záložky z SQL dotazy nebo zobrazené obsahy tabulek. Tyto záložky jsou zobrazeny ve velmi výkonném ovládacím prvku *SQLTabControl*, který je odvozen od standardního ovládacího prvku *TabControl*.

3.2.4.1 Typy záložek v ovládacím prvku *SQLTabControl*

Každý panel zobrazený ve středním panelu je reprezentován formou záložky ovládacího prvku *SQLTabControl*. Název záložky je jméno otevřeného nebo uloženého souboru u záložky typu SQL dotaz, nebo je reprezentován složením dvou slov a to „TBL - + *název tabulky*“ u záložky s otevřeným obsahem tabulky.

Kontextové menu každé záložky obsahuje tyto funkce:


- **Save All Queries** - Uloží všechny neuložené záložky typu SQL dotaz, ty, které ještě nebyly ukládány, budou uloženy pod jménem souboru, definovaným uživatelem
- **Close** - Zruší tuto záložku (u neuložených záložek typu SQL dotaz se aplikace dotáže na uložení této záložky)
- **Close All** - Zruší všechny záložky (u neuložených záložek typu SQL dotaz se aplikace dotáže na uložení jednotlivých záložek tohoto typu)
- **Close All but This** - Zruší všechny záložky, kromě právě vybrané záložky (u neuložených záložek typu SQL dotaz se aplikace dotáže na uložení jednotlivých záložek tohoto typu)
- **Options** - Otevře formulář s nastavení aplikace


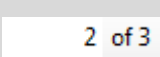




3.2.4.2 Editace záznamů tabulky

Pokud stiskneme pravé tlačítko myši na názvu tabulky a z kontextového menu zvolíme položku *Open Table*, zobrazí se její obsah v panelu nové záložky. Pro zobrazení obsahu tabulky je využit ovládací prvek *DataGridView*, který poskytuje opravdu velmi výkonné uživatelské rozhraní pro prohlížení a editaci záznamů v tabulkové podobě.

V horní části panelu s obsahem tabulky je nástrojová lišta, sloužící pro pohyb mezi zobrazenými záznamy tabulky. Popis jednotlivých tlačítek nástrojové lišty je v Tabulce 13.

Tabulka 13: Popis tlačítek pro editaci záznamů. Zdroj [Vlastní zpracování]

Tlačítko nástr. lišty	Popis funkce
	Přesune kurzor na první záznam v tabulce

	Přesune kurzor o jeden řádek zpět
	Přesune kurzor na zadané číslo záznamu v tabulce, v pravé části je zobrazena informace, kolik záznamů tabulka obsahuje celkem
	Přesune kurzor o jeden řádek dopředu
	Přesune kurzor na poslední řádek v tabulce
	Přesune kurzor na pozici vkládání nového záznamu do tabulky
	Smaže aktuální záznam z tabulky

Pod nástrojovou lištou se nachází ovládací prvek *DataGridView*, ve kterém je možné editovat hodnoty tabulky. Pro pohyb mezi buňkami tabulky je možné použít kurzorové klávesy, nebo levé tlačítko myši. Sloupec, který je definován jako **Primární klíč**³², má světle modrou barvu pozadí a v prvním řádku tabulky, kde jsou zobrazeny názvy sloupců, je značka PK. Pokud je některý ze sloupců definován jako **Identity**³³, je v názvu tohoto sloupce značka RO (Read Only).

V tabulce je možné měnit všechny hodnoty kromě hodnot ve sloupci se značkou RO. Při editaci jednoho záznamu (řádku) se pohybujeme mezi buňkami pomocí kurzorových šipek „doleva“ a „doprava“. Pro potvrzení změn v editovaném záznamu je zapotřebí stisknout klávesu „Enter“ nebo stisknout kurzorové klávesy „nahoru“, nebo „dolu“. Poté se pokusí aplikace aktualizovat Vámi editovaný záznam, pokud vše proběhne v pořádku, kurzor opustí editovaný řádek. V opačném případě vypíše aplikace chybové hlášení a je zapotřebí tuto chybu opravit, nebo stisknutím klávesy „Escape“ zrušit změny v editovaném řádku. Buňky, ve kterých se změnila jejich hodnota, mají růžovou barvu (nebo jinou, pokud jsme si v nastavení aplikace tuto barvu změnili) a po úspěšném uložení změn se opět

³² PK - Primární klíč, je typ sloupce, který má vždy unikátní hodnotu a často se využívá pro definici vazeb (relací) mezi jinými tabulkami (entitami)

³³ Pokud je sloupec definován jako Identity, jsou jeho hodnoty vypočítávány jako navyšující se číslo o 1, používá se pro automatické číslování záznamů, nejčastěji ve spojení s Primárním klíčem

jejich barva změní na původní. Poslední řádek má speciální význam a využívá se ke vkládání nových záznamů do tabulky. Pokud nastane chyba při vkládání nového záznamu, nebo požadujeme zrušit proces vložení nového záznamu, opět lze stisknout klávesu „Escape“ a nově vkládaný záznam bude zrušen.

3.2.4.3 Formulování textu SQL dotazů

SQL dotazy lze formulovat pomocí textu v záložkách typu SQL dotaz. Novou záložku pro psaní SQL dotazu vytvoříme pomocí tlačítka *New Query*, nebo klávesovou zkratkou „Ctrl + N“. V nově vytvořené záložce lze psát formou textu SQL dotazy. Syntaxe jednotlivých elementů SQL dotazu je při každé změně textu kontrolována, a pokud je rozpoznáno slovo definované standardem SQL, je následně obarveno rozdílnou barvou. Zvýrazňování SQL elementů závisí na nastavení aplikace ve formuláři *Options*. V nastavení aplikace lze nastavovat, zdali mají být zvýrazňovány elementy jazyka SQL a jakou mají mít barvu, což je důležitá vlastnost pro uživatele, kteří jsou zvyklí na určité barvy z předešlých aplikací. Textové pole záložky pro formulování SQL dotazů je standardní objekt operačního systému Windows a lze s ním pracovat stejným způsobem jako s jinými ovládacími prvky pro vkládání a editaci textu. Ve spodní části textového pole pro formulaci SQL dotazů je tzv. *StatusBar*, zobrazující stav vykonávání SQL dotazů v levé části a pozici textového kurzoru v pravé části.

Vykonání SQL dotazů obsažených ve vybrané záložce se provede pomocí tlačítka *Run Query*, nebo klávesovou zkratkou „F5“.

Při formulování textu SQL dotazů, je možné využít funkci, která je v současné verzi aplikace *SQLCommander* pouze pro testovací účely, která má název **IntelliSense**. Jedná se o funkci automatického vkládání klíčových slov do textu SQL dotazů. Po napsání několika prvních znaků klíčového slova stiskneme kombinaci kláves „Ctrl + Mezerník“ a pokud aplikace nalezne klíčové slovo, které začíná na tyto znaky tak jej automaticky vloží do textu. Pokud z vložených znaků není žádné klíčové slovo rozpoznáno, nebo je klíčových slov, které začínají těmito znaky více, zobrazí se okno s nabídkou klíčových slov. V tomto okně se pohybujeme pomocí klávesových šipek „nahoru“ a „dolu“.

Po zvolení klíčového slova stiskneme tlačítko „Enter“ a toto klíčové slovo se automaticky vloží do textu SQL dotazu a skryje se okno s klíčovými slovy.

3.2.4.4 Zobrazení výstupních dat

Po vykonání SQL dotazů, se ve spodní části středního panelu zobrazí výsledek zpracovaných dotazů. Ten je zobrazen pomocí panelů v záložkách. Poslední záložka vždy zobrazuje seznam hlášení o vykonání každého SQL dotazu. Ostatní záložky obsahují *DataSet*³⁴, zobrazený v tabulkové formě. Pro každý vykonaný SQL dotaz, který vrací *DataSet*, se vytvoří nová záložka s výstupními daty.

3.2.5 Pravý panel aplikace

Obsahuje panel Properties a Templates. V této verzi aplikace se využívá z části panel Properties a panel Templates je nevyužit, jeho funkcionality není doprogramována.

Panel Properties - Zobrazuje vlastnosti vybraného SQL připojení a jejich hodnoty. Hodnoty zobrazených vlastností nelze měnit a jsou pouze informativní. Zobrazení vlastností zajišťuje ovládací prvek *PropertyGridView*, který byl popsán v kapitole 3.2.1. *Připojení k RDBMS, část Záložka Expert Settings*. V další verzi aplikace *SQLCommander* bude zobrazovat ovládací prvek *PropertyGridView* vlastnosti většiny vybraných objektů aplikace.

Panel Templates - Tento panel není v současné verzi aplikace *SQLCommander* využit. V další verzi aplikace *SQLCommander* bude obsahovat ovládací prvek *TreeView*, který bude pomocí položek přehledně zobrazovat šablony SQL dotazů, které bude možné využít při formulování textu SQL dotazů.

³⁴ DataSet obsahuje data vyžádaná SQL dotazem

4 POROVNÁNÍ APLIKACE S ALTERNATIVAMI

Pokud se pokusím o porovnání aplikace *SQLCommander* s jinými alternativami, je zapotřebí možné alternativy rozdělit v první řadě na *komerční* a *volně dostupné*. Mezi komerčními aplikacemi pro údržbu a správu databázových systémů, které zvládnou pracovat s rozličnými databázemi různých výrobců lze řadit aplikaci SQL Edge.

4.1 SQL Edge

Tato komerční aplikace od společnosti *Bay Breeze Soft* je obdobou aplikace *SQLCommander*, s tím rozdílem, že podporuje v poslední verzi více výrobců databázových systémů a dále poskytuje podstatně větší množství speciálních funkcí. Na druhou stranu je zapotřebí přihlížet k faktu, že aplikace *SQLCommander* je pouze v testovací verzi a že aplikace SQL Edge je komerční produkt, na jehož vývoji se podílí desítky profesionálů.

4.2 Epictetus

Tato aplikace od společnosti *Antilogic Software* je z rodiny nekomerčních, volně šiřitelných aplikací. Epictetus je v současné době dostupný v testovací verzi, stejně jako aplikace *SQLCommander*. Přesto i tato testovací verze poskytuje relativně velké množství funkcí a podporuje většinu výrobců databázových systémů. Nevýhodou aplikace Epictetus je, že k jejímu spuštění je zapotřebí mít v operačním systému nainstalovaný **Java Runtime**³⁵ od společnosti *Sun Microsystems*. Toto je ovšem na druhou stranu i výhoda, protože lze aplikaci Epictetus spustit nejenom na operačních systémech Windows, ale i na operačních systémech Linux nebo MacOS. Aplikaci Epictetus bych hodnotil, jako největší konkurent aplikace *SQLCommander* a to z několika důvodů:

³⁵ Java Runtime je balíček knihoven potřebný pro spuštění aplikací vytvořených technologií Java

1. Je volně šiřitelná stejně jako aplikace *SQLCommander*
2. Umožňuje práci s rozličnými databázemi různých výrobců
3. Je to multiplatformní³⁶ typ aplikace

4.3 Shrnutí

Jednou z důležitých vlastností aplikace *SQLCommander* je také vlastnost, že ji není zapotřebí instalovat a zároveň nevyžaduje externí knihovny funkcí jako je např. **ADO**, které potřebují ke své funkci instalaci balíčku, který umožní přístup k databázovému systému daného výrobce. Tato knihovna se nazývá **DataConnector**. [9]

Myslím si, že pokud se na vývoji aplikace *SQLCommander* bude i nadále pokračovat, doplní se další funkcionalita a podpora ostatních výrobců databázových systémů, bude tato aplikace konkurence schopná jak volně šiřitelným produktům jako je aplikace Epictetus, tak i komerčním produktům jako je aplikace SQL Edge. Je velmi důležité při dalším vývoji aplikace *SQLCommander* vycházet z návrhu těchto dvou aplikací, které mohou svým způsobem sloužit jako předloha.

³⁶ Aplikaci lze spustit na rozličných typech operačních systémů

ZÁVĚR

Cílem této diplomové práce bylo vytvoření aplikace, která bude schopna provádět správu a údržbu různých databází od rozličných výrobců databázových systémů, za použití moderního programovacího jazyka. Samozřejmě bylo zapotřebí mít rozsáhlé znalosti v technologii SQL, objektově orientovaném programování a také bylo zapotřebí mít alespoň rámcovou představu o pracovním prostředí *NET Framework*, ve kterém se aplikace *SQLCommander* spouští, protože celá aplikace byla naprogramována v programovacím jazyku C#, který byl vyvinut jako programovací jazyk právě pro toto pracovní prostředí.

Výsledkem této diplomové práce je aplikace *SQLCommander*, která umožňuje správu a údržbu databázových systémů, přehledným způsobem zobrazuje seznam databází a tabulek i s jejich obsahem. Dále pak tato aplikace umožňuje psaní SQL dotazů na připojený databázový systém a přehledné zobrazení výsledků těchto dotazů v tabulkové formě. Grafické prostředí aplikace je navrženo tak, aby vyhovovalo běžným standardům pracovního prostředí Windows. Aplikace umožňuje připojení až 128 databázových systémů a lze v ní zobrazit až 128 záložek s obsahem tabulek, nebo s textem formulace SQL dotazů. Využití této aplikace předpokládám především pro její jednoduché ovládání, možnosti práce na více databázových systémech v jedné aplikaci a z důvodu inteligentního zvýrazňování textu SQL dotazů. Neopomenutelnou výhodou této aplikace vůči komerčním produktům je fakt, že tato aplikace je volně šiřitelná (*Freeware*) pod licencí GNU GPL³⁷. Je zřejmé, že aplikací tohoto typu je velmi málo a v tomto směru mají společnosti velké možnosti při vývoji softwaru, který mohou uvolňovat buď jako komerční nebo jako volně šiřitelný produkt. V každém případě pevně věřím, že výsledná aplikace *SQLCommander* bude každému správci nebo návrháři databázových systémů kvalitním nástrojem pro správu a údržbu a tato diplomová práce bude přínosem v oboru správy a údržby relačních databázových systémů (RDBMS).

³⁷ Je to licence, se kterou může být software volně šiřitelný

ZÁVĚR V ANGLIČTINĚ

The goal of this thesis was creating a tool in a modern programming language capable of maintaining various databases from different vendors. It was necessary to be able to communicate with various database managers using the SQL language, Object-oriented programming and knowing the concepts of the NET Framework because the SQL Commander was written in C#, which runs within the NET Framework.

SQLCommander is a tool for maintaining databases using a GUI that is common in the Microsoft Windows environment. It features a list of the databases, tables with their content, and querying the database systems while displaying the results in a table format. The SQL Commander allows for 128 simultaneous database connections and is capable of displaying up to 128 tabs of tables or SQL commands. This application allows easy manipulation of multiple database systems enhanced by smart SQL syntax highlighting. The advantage of the SQL Commander over commercial products is that this product is distributed under the GNU-GPL license. There is a small amount of applications offering this functionality. The SQL Commander is an alternative to commercial products and will be useful for maintaining Relational Database Management Systems (RDBMS)

SEZNAM POUŽITÉ LITERATURY

- [1] Groff, J. M. *SQL kompletní průvodce*. Brno: Computer Press, 2005. ISBN 80-251-0369-2
- [2] Molinaro, A. *SQL – Kuchařka programátora*. Brno: Computer Press, 2009. ISBN 978-80-251-2617-2
- [3] Agarwal, V. V. *Databáze v C# 2008, Průvodce programátora*. Brno: Computer Press, 2009. ISBN 978-80-251-2309-6
- [4] Gunnerson, E. *Začínáme programovat v C#*. Brno: Computer Press, 2001. ISBN 80-7226-525-3
- [5] Robinson, S. *C# Programujeme profesionálně*. Brno: Computer Press, 2003. ISBN 80-251-0085-5
- [6] Sharp, J. *Microsoft Visual C# 2010*. Brno: Computer Press, 2010. ISBN 978-80-251-3147-3
- [7] Goyvaerts, J, Levithan, S. *Regulární výrazy – Kuchařka programátora*. Brno: Computer Press, 2010. ISBN 978-80-251-1935-8
- [8] Prata, S. *Mistrovství v C++*. Brno: Computer Press, 2004. ISBN 80-251-0098-7

Ostatní zdroje

- [9] WWW.MYSQL.COM Článek týkající se datového konektoru pro prostředí NET. Dostupné z <http://dev.mysql.com/downloads/connector/net/5.2.html>
- [10] EN.CSHARP-ONLINE.NET Články týkající se Regulárních výrazů v C#. Dostupné z [http://en.csharp-online.net/CSharp Regular Expression Recipes](http://en.csharp-online.net/CSharp%20Regular%20Expression%20Recipes)
- [11] MSDN.MICROSOFT.COM Referentská příručka k MSSQL. Dostupné z <http://msdn.microsoft.com/en-us/library/ms189826.aspx>
- [12] WWW.FAQS.COM Články týkající se historie a standardu SQL. Dostupné z <http://www.faqs.org/docs/ppbook/c1164.htm>
- [13] EN.WIKIPEDIA.ORG Článek týkající se Historie a současnosti jazyka SQL. Dostupné z <http://en.wikipedia.org/wiki/SQL>
- [14] WWW.CODEPROJECT.COM Článek týkající se návrhu textového pole se zvýrazněnou syntaxí. Dostupné z <http://www.codeproject.com/KB/edit/SyntaxHighlighting.aspx>
- [15] EN.WIKIPEDIA.ORG Článek týkající se Pracovního prostředí .NET Framework od společnosti Microsoft. Dostupné z http://en.wikipedia.org/wiki/.NET_Framework
- [16] WWW.MICROSOFT.COM Kompletní popis SQL knihovny funkcí pro programátora. Dostupné z <http://msdn.microsoft.com/library/default.aspx>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

3VL	3 values
ANSI	American National Standards Institute
API	Application Programming Interface
CLR	Common Language Runtime
COM	Component Object Model
DCL	Data Control Language
DDL	Data Definition Language
DLL	Dynamic-link Library
DML	Data Manipulation Language
GUI	Graphical User Interface
IBM	International Business Machines Corporation
ISO	International Organization for Standardization
JIT	Just In Time
MDI	Multi Document Interface
MSDN	Microsoft Development Network
MSIL	Microsoft Intermediate Language
NIST	National Institute of Standards and Technology
OLE	Object Linking and Embedding
OOP	Objektově Orientované Programování
OS	Operační Systém
SQL	Structured Query Language
XML	eXtensible Markup Language

SEZNAM OBRÁZKŮ

Obrázek 1: Logo pracovního prostředí .NET Framework, Zdroj [15].....	35
Obrázek 2: Zpracování dvou vláken a) jeden procesor b) dva procesory	46
Obrázek 3: Zvýrazňování syntaxe jazyka SQL. Zdroj [Vlastní zpracování]	48

SEZNAM TABULEK

Tabulka 1: Seznam SQL standardů. Zdroj [13]	18
Tabulka 2: Výstup SQL dotazu počtu satelitů. Zdroj [Vlastní zpracování].....	22
Tabulka 3: Seznam procedurálních vylepšení jazyka SQL. Zdroj [13]	27
Tabulka 4: Popis záložky <i>File</i> v hlavním menu. Zdroj [Vlastní zpracování].....	52
Tabulka 5: Popis záložky <i>Edit</i> v hlavním menu. Zdroj [Vlastní zpracování].....	53
Tabulka 6: Popis záložky <i>View</i> v hlavním menu. Zdroj [Vlastní zpracování]	54
Tabulka 7: Popis záložky <i>Tools</i> v hlavním menu. Zdroj [Vlastní zpracování]	54
Tabulka 8: Popis záložky <i>Window</i> v hlavním menu. Zdroj [Vlastní zpracování]	55
Tabulka 9: Popis záložky <i>Help</i> v hlavním menu. Zdroj [Vlastní zpracování].....	55
Tabulka 10: Popis tlačítek nástrojové lišty DB Editor. Zdroj [Vlastní zpracování].....	56
Tabulka 11: Popis tlačítek nástrojové lišty <i>Standard</i> . Zdroj [Vlastní zpracování].....	56
Tabulka 12: Popis tlačítek nástrojové lišty levého panelu. Zdroj [Vlastní zpracování]	58
Tabulka 13: Popis tlačítek pro editaci záznamů. Zdroj [Vlastní zpracování].....	59

SEZNAM PŘÍLOH

- P I: Formulář New Connection
- P II: Editace záznamů v tabulce
- P III: Výsledky SQL dotazů
- P IV: Formulování SQL dotazů
- P V: Příklad zápisu jazyka C#
- P VI: Příklad zápisu jazyka MSIL

PŘÍLOHA P I: FORMULÁŘ NEW CONNECTION

New Connection

SQLCOMMANDER

Login Connection specifications Expert settings

SQL Server

Provider : MSSQL Server

Server name : localhost\sqlexpress

Authentication : Windows Authentication

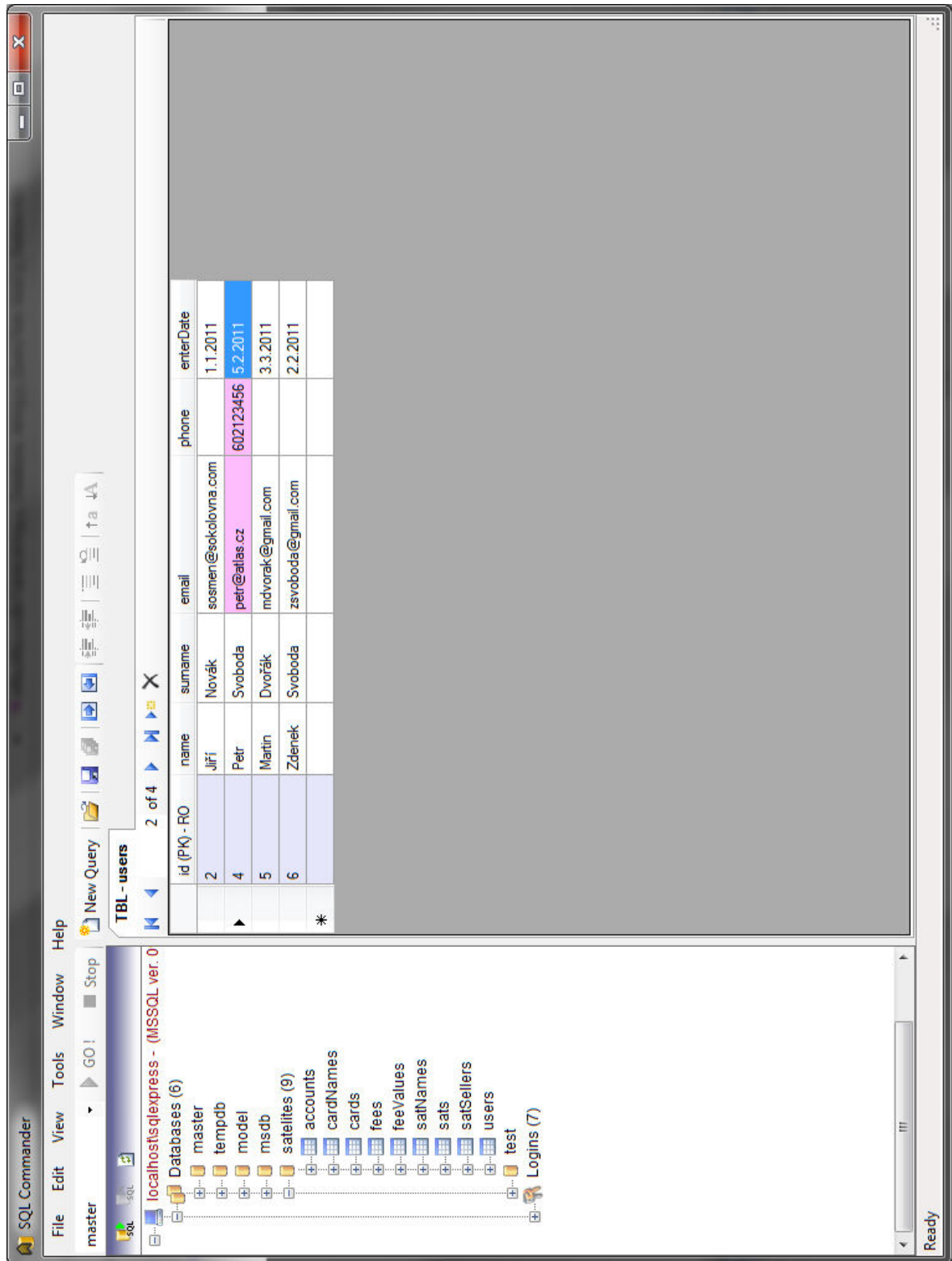
User name :

Password :

Remember password

Connect Close Help

PŘÍLOHA P II: EDITACE ZÁZNAMŮ V TABULCE



PŘÍLOHA P III: VÝSLEDKY SQL DOTAZŮ

The screenshot shows the SQL Commander interface. The top pane displays the following SQL code:

```
TBL - users
-- Zmena databaze
USE satellites
-- Dotaz na vsechny zaznamy v tabulce accounts
SELECT * FROM accounts ORDER BY login
-- Dotaz na vsechny zaznamy v tabulce users
SELECT * FROM users ORDER BY id DESC
-- Dotaz s podmínou WHERE
SELECT * FROM feeValues WHERE (fee >= 500) AND (fee <= 10000)
-- Zkouska konce dotazu
```

The bottom pane shows a tree view of the database structure for 'localhost:sqlserver - (MSSQL ver. 09.00.4053)'. The 'users' table is selected, showing its schema:

- id [PK, int, not null]
- name [varchar(50), not null]
- surname [varchar(50), not null]
- email [varchar(50), not null]
- phone [varchar(20), null]
- enterDate [datetime, null]

The results grid shows the following data:

id	name	surname	email	phone	enterDate
6	Zdenek	Svoboda	zsvoboda@gmail...	420	2.2.2011
5	Martin	Dvořák	mdvorak@gmail...	420	3.3.2011
4	Petr	Svoboda	petr@atlas.cz	602123456	5.2.2011
2	Jiří	Novák	sosmen@sokolo...		1.1.2011

The interface also shows the status 'Executing queries done' and 'Row: 13 | Col: 24'.

PŘÍLOHA P IV: FORMULOVÁNÍ SQL DOTAZŮ

The screenshot displays the SQL Commander interface. The top pane shows a tree view of the database schema for 'localhost\sqlserver - (MSSQL ver. 09.00.4053)'. The 'Databases (6)' folder is expanded, showing 'master', 'tempdb', 'model', 'msdb', and 'satellites (9)'. The 'satellites (9)' folder is further expanded, showing tables: 'accounts', 'cardNames', 'cards', 'fees', 'feeValues', 'satNames', 'sats', 'satSellers', and 'users (6)'. The 'users (6)' table is selected, showing its structure: 'id [PK, int, not null]', 'name [varchar(50), not null]', 'surname [varchar(50), not null]', 'email [varchar(50), not null]', 'phone [varchar(20), null]', and 'enterDate [datetime, null]'. The bottom pane shows a SQL script with the following content:

```
/*
Created          21.3.2010
Modified        19.8.2010
Project         Satellites
Model           MainSatModel
Company         UTB FAI Zlín
Author          sosmen
Version         1.0
Database        MS SQL 2000
*/
create database satellites
go
use satellites;
go
Create table [cards]
(
    [id] Integer Identity(1,1) NOT NULL, UNIQUE ([id]),
    [cardName] Integer NOT NULL,
    [account] Integer NOT NULL,
    Primary Key ([id])
)
go
select
```

The right side of the interface shows a query editor with a preview of the results for the 'select' statement. The preview shows a table with columns 'session_user', 'set', and 'setuser'. The status bar at the bottom right indicates 'Row: 26 | Col: 3'.

PŘÍLOHA P V: PŘÍKLAD ZÁPISU JAZYKA C#

```
public partial class SplashScreen : Form
{
    public static SplashScreen frmSplashScreen = null;
    private static Thread splashThread = null;

    // Time variables
    private double opacityIncrement = .1;
    private double opacityDecrement = .1;
    private const int TIMER_SPLASH_DELAY = 4000;
    private const int TIMER_INTERVAL = 35;

    private string statusString = "Loading app...";

    public SplashScreen()
    {
        InitializeComponent();

        this.ClientSize = this.BackgroundImage.Size;
        this.Opacity = .0;
        this.timerFading.Interval = TIMER_INTERVAL;
        this.timerFading.Start();
    }

    // Create new instance od SplashScreen a show splash
screen
    private static void ShowSplash()
    {
        frmSplashScreen = new SplashScreen();
        Application.Run(frmSplashScreen);
    }

    // Close splash screen
    public static void CloseSplash()
    {
        if (frmSplashScreen != null)
        {
            Thread.Sleep(TIMER_SPLASH_DELAY);
            // Make it start going away.
            frmSplashScreen.opacityIncrement = -
frmSplashScreen.opacityDecrement;
        }
        splashThread = null; // we do not need these any
more.
        frmSplashScreen = null;
    }
}
```

```

// Run new thread with splash screen
static public void RunThreadSplashScreen()
{
    // Make sure it is only launched once.
    if (frmSplashScreen != null)
        return;
    splashThread = new Thread(new
ThreadStart(SplashScreen.ShowSplash));
    splashThread.IsBackground = true;
    splashThread.SetApartmentState(ApartmentState.STA);
    splashThread.Start();
}

// Tick event
private void timerFading_Tick(object sender, EventArgs e)
{
    this.lblStatus.Text = this.statusString;

    if (opacityIncrement > 0)
    {
        if (this.Opacity < 1)
            this.Opacity += opacityIncrement;
    }
    else
    {
        if (this.Opacity > 0)
            this.Opacity += opacityIncrement;
        else
            this.Close();
    }
}

// A static method to set the status.
public static void SetStatus(string p_NewStatus)
{
    if (frmSplashScreen != null)
        frmSplashScreen.statusString = p_NewStatus;
}
}
}

```

PŘÍLOHA P VI: PŘÍKLAD ZÁPISU JAZYKA MSIL

```
.method public hidebysig specialname rtspecialname
    instance void .ctor() cil managed
{
    // Code size      86 (0x56)
    .maxstack 4
    .locals init ([0] class
        [System.Windows.Forms]System.Windows.Forms.Button btn)
    IL_0000: ldarg.0
    IL_0001: call      instance void
        [System.Windows.Forms]System.Windows.Forms.Form::.ctor()
    IL_0006: ldarg.0
    IL_0007: ldstr
        bytearray (4A 00 65 00 64 00 6E 00 6F 00 64 00 75 00 63 00
            68 00 E9 00 20 00 74 00 6C 00 61 00 0D 01 ED 00
            74 00 6B 00 6F 00 )
    IL_000c: callvirt instance void
        [System.Windows.Forms]System.Windows.Forms.Control::
        set_Text(string)
    IL_0011: ldarg.0
    IL_0012: newobj     instance void
        [mscorlib]System.Random::.ctor()
    IL_0017: stfld     class
        [mscorlib]System.Random
OvladaciPrvky.FormJednoducheTlacitko::
    rand
    IL_001c: newobj     instance void
        [System.Windows.Forms]System.Windows.Forms.Button::
        .ctor()
    IL_0021: stloc.0
    IL_0022: ldloc.0
    IL_0023: ldstr     "Klikni!"
    IL_0028: callvirt instance void
        [System.Windows.Forms]System.Windows.Forms.Control::
```

```

        set_Text(string)
IL_002d:  ldloc.0
IL_002e:  ldc.i4.s   20
IL_0030:  ldc.i4.s   20
IL_0032:  newobj     instance void
        [System.Drawing]System.Drawing.Point::
        .ctor(int32, int32)
IL_0037:  callvirt  instance void
        [System.Windows.Forms]System.Windows.Forms.Control::
        set_Location(valuetype
[System.Drawing]System.Drawing.Point)
IL_003c:  ldloc.0
IL_003d:  ldarg.0
IL_003e:  callvirt  instance void
        [System.Windows.Forms]System.Windows.Forms.Control::
        set_Parent(class [Systemem.Windows.Forms]
        System.Windows.Forms.Control)
IL_0043:  ldloc.0
IL_0044:  ldarg.0
IL_0045:  ldftn     instance void
        OvladaciPrvky.FormJednoducheTlacitko::
        btn_Click(object, class [mscorlib]System.EventArgs)
IL_004b:  newobj     instance void
        [mscorlib]System.EventHandler::
        .ctor(object, native int)
IL_0050:  callvirt  instance void
        [System.Windows.Forms]System.Windows.Forms.Control::
        add_Click(class [mscorlib]System.EventHandler)
IL_0055:  ret
} // end of method FormJednoducheTlacitko::.ctor

```