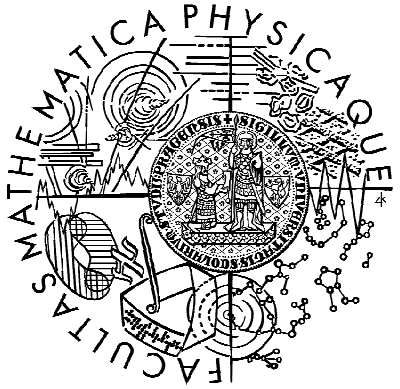
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Ondřej Dolejš

Databázový manažer

Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. Filip Zavoral, Ph.D.
Studijní program: Informatika
Studijní obor: Správa počítačových systémů

2006

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu bakalářské práce panu RNDr. Filipu Zavoralovi, Ph.D. za pozitivní a především podnětný přístup při vedení a směrování vývoje mé práce.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 1.8.2006. Ondřej Dolejš

Obsah

Název práce: Databázový manažer
Autor: Ondřej Dolejš
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Filip Zavoral, Ph.D.
e-mail vedoucího: Filip.Zavoral@mff.cuni.cz

Abstrakt: Předložená práce pojednává o mnou vytvořeném programu DBManager. Práce sleduje důvody a motivace pro jeho vznik, popisuje analýzu předcházející vzniku programu stejně jako jeho implementaci a výslednou funkčnost. Program DBManager je lehký databázový manažer schopný připojit se na databázi Oracle a Microsoft SQL server. Program obsahuje SQL konzoli s výstupem do tabulky a spolupracující prohlížeč databázových objektů. Výsledky dotazů jdou třídit, filtrovat a upravovat, načež lze upravená data jednoduše poslat zpět do databáze. Prohlížeč databázových objektů přehledně zobrazuje běžné databázové objekty, u těch nejběžnějších, tedy u tabulek a pohledů, i s možností přidávání a úpravy pomocí uživatelsky přívětivého průvodce. SQL dotazy i výsledky dotazů lze ukládat i načítat ze souboru.

Klíčová slova: databáze, manažer, SQL, lehký,

Title: Database manager
Author: Ondřej Dolejš
Department: Department of Software Engineering
Supervisor: RNDr. Filip Zavoral, Ph.D.
Supervisor's e-mail address: Filip.Zavoral@mff.cuni.cz

Abstract: The presented work is about the program DBManager, which I created. This work follows the reasons and motivations which had led to creation of the program and describes analysis preceding the creation of the program. Furthermore it describes an implementation of the program and its resulting functionality. The program DBManager is a lightweight database manager with ability to connect to the Oracle database as well as to the MS SQL Server database. The program contains the SQL console with output into the table and cooperating object browser. Results can be filtered, sorted and modified, and the modified data can be easily sent back to the database. The Object browser shows us well-arranged common database objects. The most usual database objects, such as tables or views, can be created or modified by using user friendly wizard. SQL commands as well as results of the commands can be saved and loaded from a file.

Keywords: database, manager, SQL, lightweight

Kapitola 1 – Úvod

1.1 Cíl

Cílem této bakalářské práce je vytvořit lehký databázový manažer. Svými funkcemi by měl pokrývat potřeby programátora tvořícího program, který pracuje s databází a potřeby správce menší databáze, kde ať už z jakýchkoliv důvodů nemá přístup k pokročilejším nástrojům pro správu. Program musí obsahovat přinejmenším SQL konzoli, aby mohl uživatel neomezeně pracovat s databází. Práce se samotnou konzolí je však vcelku nepohodlná, a tak bude mým hlavním cílem tuto práci uživateli maximálně zpříjemnit.

1.2 Cesty k cíli

Zkusme si představit, co by našemu uživateli, kterého jsme si v kapitole 1.1 určili, učinilo práci pohodlnější.

Prvním zpříjemněním určitě bude možnost nahlížet do systémových tabulek během psaní příkazů. Uživatel si nemůže pamatovat identifikátory všech objektů databáze, a tak dozajista ocení strukturovaný přehled alespoň těch nejpoužívanějších. Nejpřehlednějším způsobem, jak objekty zobrazit budou zřejmě tabulky, a vzhledem k možnému množství objektů v databázi by měla být samozřejmostí možnost třídění podle vybraného parametru. V programech podobných mému se tomuto nástroji většinou říká „Object browser“, či česky prohlížeč (databázových) objektů, budu se tedy tohoto označení držet. Dále by bylo vhodné, pokud už uživatel hledaný identifikátor objektu v Object Browseru najde, aby jeho název nemusel opisovat ručně.

Dalším zpříjemněním by mohlo být, kdyby výsledná data nebyla vypisována textově zpět do konzole. Daleko přehlednější výstup by představovala tabulka. Rozsah výsledných dat může být značný, a tak by opět měla být samozřejmá možnost třídění. Šikovnou pomůckou by pak také mohla být možnost filtrování výsledných dat bez nutnosti připisování WHERE [podmínky] za dotazy.

Další urychlení práce by mohlo přinést zjednodušené vytváření a upravování struktury objektů a dat v nich obsažených. Například při vytváření a úpravě tabulek se dost klíčových slov opakuje, a tak by šlo tyto činnosti automatizovat a uživatele jich ušetřit. Nabízí se možnost průvodce, kde by uživatel vyplňoval jen opravdu potřebná data a syntaxe příkazu by z nich byla vytvořena automaticky. Je samozřejmě třeba vybrat jen objekty, u kterých průvodce má smysl a přinese kýžené zrychlení práce. Podobnou filozofií by se mělo řídit i upravování obsahu objektů – oprostit uživatele od zbytečného psaní. Zde se zdá nejjednodušší přepisovat data přímo v tabulce s výsledky, kde má uživatel kontext se zbytkem dat.

A nakonec zbývá spousta příjemných maličkostí – program by si kupříkladu mohl pamatovat zadaná spojení na databáze, aby uživatel nemusel při každém spuštění programu vypisovat, často ne příliš mnemotechnické, připojovací údaje. Taktéž by uživatel jistě ocenil, kdyby si mohl svoje SQL příkazy, stejně jako tabulku s výsledky, uložit a načíst ze souboru na disku. Stojíme zde před volbou, zda data na disku ukládat ve formě textového souboru, či netextového souboru s vlastní strukturou. Textový soubor je však jasnou volbou už kvůli kompatibilitě s ostatními programy (uloženou tabulku pak lze kupříkladu otevřít v aplikaci MS Excel) a možnosti prohlížet si uložené soubory jakýmkoliv dostupným textovým editorem.

1.3 Motivace

Důvodů, proč jsem si vybral pro svou bakalářskou právě toto téma, je více. Jedním z nich byla určitě touha proniknout hlouběji do jazyka SQL a světa databází, ke kterým jsem si během přednášek na této fakultě vytvořil pozitivní vztah. A druhým důvodem byla poptávka po takovémto programu, kterou vyslovil můj otec. Stal se tak pro mě impulzem k tvorbě a zároveň také, dalo by se říci, koncovým zákazníkem.

1.4 Přehled dalších kapitol

Druhá kapitola se zabývá analýzou a návrhem aplikace. V této kapitole popisuji možnosti, jak vytyčený cíl naplnit a důvody pro rozhodnutí, která musela být učiněna ještě než se začne s implementační fází.

Třetí kapitola se zabývá implementační fází vývoje aplikace. Vysvětluje vnitřní fungování programu včetně rozhodnutí, která byla učiněna během implementační fáze

Ve čtvrté kapitole jsou stručně popsané jednotlivé dokumenty, které byly vytvořeny k programu DBManager a jsou jeho nedílnou součástí.

Závěrečná kapitola se zabývá programem DBManager z nadhledu. Nabízí jak „pohled do minulosti“, kde hodnotí práci a splnění cílů, tak i „pohled do budoucnosti“, kde nastiňuje možnosti dalšího vývoje aplikace.

Práci uzavírá kapitola s přílohami, ve které nalezneme specifikaci a příručky k programu popisované ve čtvrté kapitole.

Kapitola 2 – Analýza a návrh

2.1 Databáze

Vzhledem k tomu, že hlavním úkolem programu DBManager je práce s databázemi, je nutné se v první řadě zabývat jimi.

2.1.1 Připojení

Než můžeme začít s vybranou databází pracovat, musíme se na ní nejdříve dokázat připojit. A to není tak samozřejmé, jak by se mohlo zdát. Jedním ze způsobů, jak se na databázi připojit, je použít knihovny pro danou databázi vytvořené autory databáze. Jelikož jsou však tyto knihovny specifické pro každou databázi a můj program by se měl umět připojovat na různé typy databází, toto zcela určitě nebude cesta, kterou bych se chtěl ubírat. Naštěstí existují univerzální rozhraní, která konektivitu na různé databáze umožňují. Zde je přehled těch nejrozšířenějších:

- **ODBC [6]**

  Open Database Connectivity (ODBC) je mezinárodní standard pro manipulaci s daty na relačních databázích za pomoci dotazů s SQL syntaxí. Výhodou je obrovské rozšíření (téměř každá relační databáze ODBC podporuje), nevýhodou pak omezení ODBC jen na relační databáze a syntaxi SQL.

- **OLE DB [7]**

  OLE DB je otevřený standard pro přístup k jakémukoliv druhu dat. Nemá žádné omezení na syntaxi příkazů a jediným omezením na data je, že musí být možné je zobrazit jako tabulku. OLE DB staví na úspěchu ODBC a také se velice rychle rozšiřuje

- **ADO [8]**

  ADO je jakousi nadstavbou OLE DB. Je to univerzální vysokoúrovňové rozhraní k datům získaným pomocí OLE DB.

Vzhledem k tomu, že ODBC je nejrozšířenějším standardem pro univerzální přístup k datům a můj program ani zdaleka nemíří za hranice jeho možností, zvolil jsem pro připojování DBManageru k databázím právě ODBC.

2.1.2 SQL a jeho syntaxe

Ačkoliv jsme již našli univerzální rozhraní pro připojení, stále ještě není v oblasti univerzálnosti vyhráno. Pro zadávání příkazů a dotazů do databáze budeme používat jazyk SQL [9]. Ten od roku 1986, kdy byl poprvé standardizován, prodělal hned několik revizí, ale přesto jeho dodržování není zdaleka precizní. Je k tomu hned několik důvodů:

- Jednou z příčin je složitost a rozsáhlost standardu SQL, díky níž většina databází neimplementuje standard celý.
- Standard velice přesně definuje syntaxi jazyka SQL, avšak sémantika už tak dobře definovaná není, a tím vzniká prostor pro víceznačnosti.
- Standard neřeší chování v celé řadě oblastí (např. indexy), ponechávaje tak rozhodnutí na implementaci.
- Některé teorie dokonce tvrdí, že odchylky od standardu a následná nekompatibilita jsou záměrem výrobců jak dostat zákazníky do situace, ve které jsou nuceni produkt používat již z toho důvodu, že přechod na jiný produkt je nemyslitelně obtížný.

Díky tomu se setkáme s množstvím specifických, vzájemně nekompatibilních rozšíření jazyka SQL. Jako příklad můžeme uvést PL/SQL firmy Oracle [10] či Transact-SQL firmy Microsoft [11].

A tak v oblasti jazyka SQL podobně univerzální řešení jako u připojení (ODBC) bohužel nenajdeme. Při vývoji aplikace jsem zkoušel, zda bych nenašel alespoň rozumnou společnou podmnožinu syntaxe SQL různých databází, avšak ani to se nepodařilo. Na základě tohoto zjištění budu muset při vytváření aplikace v určitých místech postupovat různě pro konkrétní databáze.

2.2 Management databáze

Jelikož cílem práce je vytvořit databázový manažer, musíme si nejdříve ujasnit, co budeme pod pojmem managementu databáze rozumět. Můj program rozhodně svými ambicemi nemíří ke správě celé databáze. Za cíl (viz 1.1) jsem si totiž určil program, který bude sloužit uživateli, jež chce znát strukturu databáze a její obsah. Tvorba programu na správu celé databáze je zcela zřejmě daleko za hranicemi možností jednoho programátora. Program DBManager se tedy zaměří jen na správu obsahu databáze – jejích schémat a dat.

2.3 Konkurenční produkty

Dobré vodítko pro to, co by měl můj program umět a na jaké místo na trhu teoreticky směřovat, nám mohou podat podobné konkurenční produkty a jejich případné nedostatky.

2.3.1 Placené produkty

Při hledání programů podobných mému jsem se na Internetu setkal se zajímavým jevem: Občas je na internetu slučován pojem freeware a trial verze. Po celou dobu je produkt označován jako freeware, ale ve skutečnosti se jedná o trial verzi. To se ovšem uživatel dozví, až když si přečte EULA (End User License Agreement). Jelikož však EULA čte mizivé procento uživatelů, čeká je překvapení teprve při zkoumání aplikace, popřípadě při vypršení zkušební doby, kdy program odmítne dál pracovat. Zde jsou programy, které jsem porovnával:

- **DB Commander [12]**

  Program se visuelně tváří trochu jako Total Commander, umí se připojit na dvě různé databáze najednou a kopírovat mezi nimi data (zdrojová a cílová tabulka nemusí být stejného typu). Program se oproti mému cíli více zaměřuje na kopírování mezi databázemi a obsahuje jen velmi jednoduchou SQL konzoli, takže to není nástroj, se kterým by se dala databáze pohodlně prohlížet. Program navíc několikrát vyvolal chybu „Access violation“.

  Program stojí 99 dolarů

- **Advanced Query Tool [13]**

Program se blíží tomu, o co bych chtěl usilovat (ačkoliv obsahuje hodně dalších možností, které zahrnout nehodlám). Obsahuje Object Browser se stromovou strukturou. Obsah databázových objektů lze zobrazit a měnit, nelze však měnit strukturu databázových objektů samotných. Druhou částí programu je intuitivní „sestavovač“ SQL dotazů s grafickým rozhraním, kde si uživatel může v podstatě celý dotaz „naklikat“.

Program stojí 150 dolarů

- **Toad freeware (lehká verze, zdarma) [14]**

Program je v principu tím, co bych chtěl vytvořit. Má SQL konzoli na vykonávání příkazů, která je podpořena prohlížečem databázových objektů. Z něj lze dvojklikem vkládat názvy tabulek či sloupců přímo do SQL příkazu, popřípadě objekty měnit, přidávat či mazat. Program je silně odlehčenou a zjednodušenou verzí plné verze Toadu, který poskytuje velice pokročilé nástroje na správu databáze. I přes věhlas tohoto nástroje jsem se však několikrát setkal s výjimkou „access violation“. Program je sice oficiálně prezentován jako freeware, ale příčí se mi zařadit ho do následující kategorie zcela bezplatných programů. V roce 2004 byla totiž takto zdarma k dispozici lehká verze Toadu pro Oracle, ovšem 1.1.2005 „vypršela“ a od té doby je naopak zdarma k dispozici lehká verze Toadu pro MS SQL server. Těžko se dá tedy odlehčený bezplatný Toad považovat za projev dobrého srdce, jde o marketingovou strategii firmy Quest Software na získání více zákazníků pro svoji plnou verzi Toadu.

Cena plné verze Toadu – 800 dolarů a více (podle verze)

2.3.2 Produkty zdarma

Placených programů je na internetu široký výběr, o poznání méně je už však programů bezplatných. Většina z nich se omezuje jen na jednoduchou konzoli, jediný rozsáhlejší, který jsem našel je:

- **SQL Tools 1.4 [15]**

  Tento program je hodně podobný tomu, jaký bych chtěl sám vytvořit. Obsahuje jednak SQL konzoli s výstupem do tabulky a zvýrazňováním klíčových slov (syntax highlighting), jednak prohlížeč objektů databáze, přehledně uspořádaných do karet s tabulkami. Objekty samotné lze prohlížet i měnit stejně jako jejich obsah, ale v podstatě jen textově. Funkcí Drag and Drop lze přenášet objekty přímo do SQL konzole a pohodlně tak získávat přehled o databázi zároveň s vyzkoušením výsledků potřebného SQL dotazu. Program je freeware, ale je určen pouze pro databázi Oracle.

Z porovnání a studie výše zmíněných programů jsem vyvodil následující závěry pro moji práci:

- Dokázat se připojit na databázi Oracle i MS SQL server není mezi existujícími programy vůbec samozřejmá vlastnost a pokud budu v programu připojení na obě nejdůležitější databáze podporovat, bude to znamenat konkurenční výhodu
- Některé programy nebyly příliš přehledné a intuitivní. Při implementaci tedy budu klást důraz na uživatelskou přívětivost programu
- Dalším silným nedostatkem zkoumaných programů byla jejich nestabilita. U svého programu tedy budu usilovat o co nejvyšší stabilitu.
- Většina programů potřebovala instalaci, což z různých důvodů nemusí být na cílovém počítači možné. Mojí snahou bude, aby program žádnou nevyžadoval.

2.4 Specifikace

Na základě analýzy potřeb a možností vznikla specifikace programu DBManager. Bude to lehký databázový manažer s SQL konzolí, která bude zpracovávat SQL dotazy a výsledky vypisovat do tabulky. S SQL konzolí bude úzce spolupracovat prohlížeč databázových objektů. Tento prohlížeč bude v tabulkách přehledně rozdělených do karet zobrazovat jednotlivé druhy databázových objektů a bude možné z něho přenášet názvy těchto objektů přímo do konzole. Data ve výsledkové tabulce bude možné třídit a filtrovat a po případné úpravě půjdou vrátit zpět do databáze. Nejběžnější databázové objekty pak budou moci být vytvářeny a upravovány pomocí uživatelsky přívětivého průvodce. Stejně tak půjde u nejpoužívanějších databázových objektů zobrazit jejich definici.

Kapitola 3 – Implementace

3.1 Výběr vývojového nástroje

Důkladný výběr příjemného a intuitivního vývojového prostředí ovlivní nejen kvalitu výsledného programu, ale také časovou náročnost celého projektu. Nejdůležitější jsou však zkušenosti programátora s daným prostředím, což bylo také hlavním důvodem pro moji volbu Visual Studia .NET 2003 společnosti Microsoft [16], se kterým jsme pracovali při cvičeních k různým předmětům fakulty. Pro tuto volbu hovořila také možnost jeho bezplatného získání v programu ELMS [17] společnosti Microsoft.

3.2 Vnější reprezentace

Hned v počátcích implementace je třeba si stanovit, jak budeme výsledky i požadavky na uživatele prezentovat. .NET [18] nabízí mnoho standardních prvků, mezi nimiž je i solidní podpora pro práci s daty. Účelem programování není „opětovné objevování kola“, a tak není důvod je nevyužít.

Základem programu je SQL konzole. Pro její realizaci se nabízí použití prvku textového pole, který se v .NETu jmenuje **RichTextBox**. Je to dostatečně flexibilní nástroj pro práci s textem, a proto najde využití i v dalších místech aplikace. Pojmenování těchto objektů identifikátorem „memo“ je jen důsledek předchozí zkušenosti s Delphi, kde se takto prvek textového pole nazývá.

Pro zobrazování výsledků jsem zvolil jako nejvhodnější prostředek tabulku. .NET obsahuje pro mé účely velice šikovný prvek **DataGrid**, který je schopen úzce spolupracovat se zdroji dat, a tak jej v aplikaci využiji.

Další částí mé aplikace je Object Browser. Jelikož se však nejedná o nic jiného než zobrazování výsledků správně položených dotazů do systémových tabulek, použiji opět **DataGrid**.

Poslední větší část programu, pro kterou hledám reprezentaci jsou průvodci. Podle mého názoru je nejlogičtější řešení pomocí nového okna, na kterém budou zobrazeny standardní ovládací prvky (tlačítka, kolonky, check boxy, …), pomocí kterých by uživatel zadával parametry vytvářeného objektu. Stojíme zde však před otázkou, jak uživateli pro jeho představu prezentovat vytvářený objekt. U průvodce zabývajícího se tabulkami se jedná pouze o prezentaci sloupců tabulky a jejich vlastnosti a omezení, a tudíž je prezentace pomocí **DataGridu** zcela dostačující. U průvodce zabývajícího se pohledy musíme kromě sloupců prezentovat i dotaz, který pohled tvoří. K tomu postačí již dříve zmíněný **RichTextBox**.

3.3 Vnitřní reprezentace

Po určení vnější reprezentace je také třeba stanovit, jak budeme prezentovaná data uchovávat uvnitř programu.

Uchovávání obsahu SQL konzole a ostatních textových polí není nutno řešit, jelikož se o to stará sama komponenta **RichTextBox**. Pro uchovávání a přístup k tabulkám získaným z databáze poskytuje .NET třídu **DataTable**. Prezentace **DataTable** pomocí **DataGrid** je velice jednoduchá, a proto jsem ji také zvolil jako vnitřní reprezentaci tabulek. Vzhledem k tomu, že program vyžaduje reprezentaci několika tabulek najednou, využil jsem ještě třídy **DataSet**, která představuje pole tabulek **DataTable**.

Objekt typu **DataSet** obsahuje každé z oken pracujících s databází. Při dotazu do databáze se do něj přidá tabulka typu **DataTable** s vrácenými daty a prezentuje se pomocí prvku **DataGrid**. Při dalším dotazu se stará tabulka **DataTable** odstraní a přidá se nová s aktuálními daty.

V úvodu implementace však také vyvstala řada požadavků, které komponenty .NETu nedokázaly pokrýt:

Ačkoliv to ve specifikaci nebylo explicitně uvedené, ukázalo se, že zadávání připojovacích údajů stále znovu je krajně nepohodlné. Bylo tedy třeba, aby si program

pamatoval zadané údaje i mezi jednotlivými spuštěními programu. Z toho také vyplynulo, že je nutné stanovit si vnitřní reprezentaci zadaných připojení, kterou budeme ukládat do konfiguračního souboru. Pro tyto účely jsem vytvořil třídy **Connection** a **Seznam**. Třída **Connection** reprezentuje nezbytné údaje jednoho připojení k databázi. Má nadefinovaný jen konstruktor, destruktor a předefinovaný operátor =, což znamená, že se k jejím položkám přistupuje přímo. Třída **Seznam** představuje seznam připojení. Obsahuje pole objektů typu **Connection**, metody pro práci s jednotlivými připojeními a automatické zvětšování pole.

Třídy **Seznam** a **Connection** jsou využity v hlavním okně aplikace, kde najdeme proměnné **Actual**, **Chosen** (instance třídy **Connection**) a **Pripoj** (instance třídy **Seznam**). V proměnné **Chosen** je udržováno připojení, které uživatel naposledy vybral ve formuláři „Nastavení připojení“. Poté co se uživatel s parametry z proměnné **Chosen** úspěšně připojí na databázi, je její obsah zkopírován do proměnné **Actual**, která uchovává parametry připojení, které bylo naposledy úspěšně navázáno. V proměnné **Pripoj** jsou udržována všechna připojení, která byla načtena z konfiguračního souboru.

Další důležitý požadavek vyvstal při tvorbě průvodců. Bylo nezbytné určit reprezentaci pro nově vytvářený či upravovaný objekt. Různé databázové objekty se však natolik liší svými potřebami reprezentace, že jsem tento problém řešil u každého druhu objektu zvlášť. Pro průvodce úpravou a vytvářením tabulky jsem vytvořil třídy **Sloupec** a **Tab**. Třída **Sloupec** reprezentuje jeden sloupec tabulky včetně jeho typu a integritních omezení. K položkám třídy se opět přistupuje přímo. Třída **Tab** reprezentuje seznam sloupců tabulky. Stejně jako třída **Connection** obsahuje metody pro práci s jednotlivými sloupci a automatické zvětšování pole. Pro průvodce úpravou a vytvářením pohledu byly vytvořeny třídy **WSloupec** a **TView**, které jsou analogické třídám **Sloupec** a **Tab**, pouze obsahují méně položek ve třídě **WSloupec**.

Třídy **Tab** a **Sloupec** jsou využity v okně „Průvodce vytvořením / úpravou tabulky“, kde najdeme proměnné **Tabulka**, **OriginalTabulka** (instance třídy **Tab**) a **pom** (instance třídy **Sloupec**). Proměnná **Tabulka** slouží k uchovávání sloupců vytvářené či upravované tabulky. Proměnná **OriginaTabulka** je využita jen v případě „Průvodce úpravou tabulky“. Při otevření okna jsou do ní načteny sloupce upravované tabulky včetně parametrů. Při vytváření příslušných ALTER příkazů v metodě **ObnovMemo** je pak

proměnná **OriginalTabulka** využita pro srovnání s proměnnou **Tabulka**. Proměnná **pom** je pomocná proměnná sloužící metodám okna k různým účelům.

Třídy **TView** a **WSloupec** jsou využity v okně „Průvodce vytvořením / úpravou pohledu“, kde najdeme proměnné **NewView** (instance třídy **TView**) a **wpom** (instance třídy **WSloupec**). Proměnná **NewView** slouží k uchovávání sloupců vytvářeného či upravovaného pohledu. Proměnná **pom** je pomocná proměnná sloužící metodám okna k různým účelům.

3.4 Připojení k databázi

Pro připojení k databázi .NET nabízí třídu **OdbcConnection**. Pro potřeby naší aplikace je plně vyhovující a proto ji použiji. V průběhu implementace však bylo zjištěno, že Visual Studio nepodporuje spojení na databáze Oracle pomocí ODBC. Vzhledem k pokročilé fázi vývoje programu jsem tedy pro připojení na Oracle zvolil standard OLE DB, o jehož použití jsem při analýze uvažoval pro celou aplikaci (viz 2.1.1). Pro připojení k databázi pomocí OLE DB nám .NET poskytuje také třídu **OleDbConnection**, kterou v programu využiji. Sadu nástrojů, jenž v programu používám pro práci s databází doplňuje třída **OdbcCommand** a **OleDbCommand**, které reprezentují SQL příkaz pro dané připojení. Získávání dat z databáze má v mém programu dvě podoby:

Pokud nepotřebuji vrácená data upravovat a jsou přímo určená k zobrazení, použiji třídu **OdbcAdapter** či **OleDbAdapter**. Ty dokáží výsledkem dotazu naplnit tabulku typu **DataTable**, kterou následně zobrazím komponentou **DataGrid**.

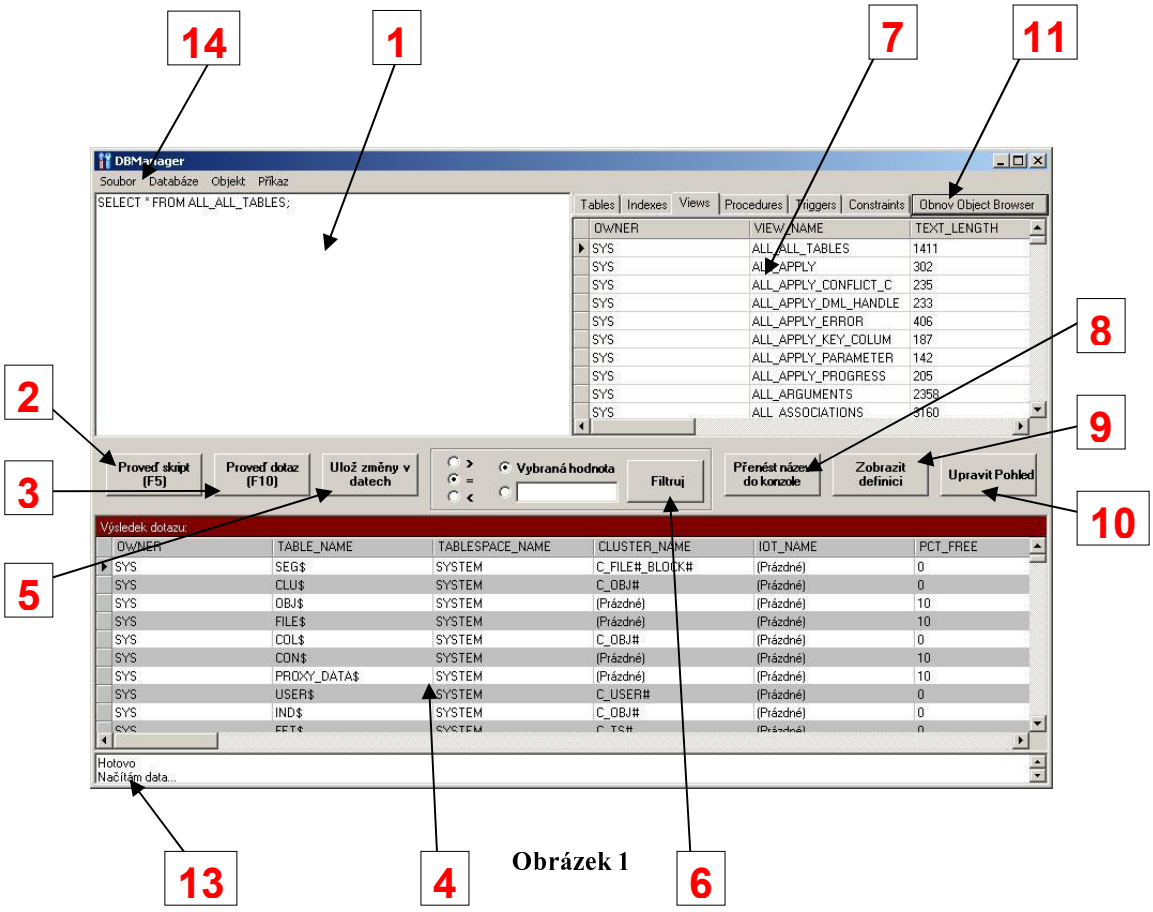
Pokud však potřebuji získaná data zpracovat, je výše zmíněné řešení nevhodné. Používám proto třídu **OdbcDataReader** či **OleDbDataReader**, které umožňují přistupovat k získaným datům řádek po řádku.

3.5 Popis hlavních prvků programu a zajištění jejich funkčnosti

V této kapitole popíši jednotlivé prvky (či skupiny prvků) programu a způsob, jakým zajišťuji jejich funkčnost. Pro názornost popisu budu využívat obrázků oken programu. Popis programu z dalších hledisek (s využitím stejných obrázků jako zde) je možné najít v Uživatelské příručce (popis viz. 4.2, obsah viz Přílohy) a Programátorské příručce (popis viz 4.3, obsah viz Přílohy).

3.5.1 Hlavní okno aplikace

Hlavní okno aplikace je vizuálně rozděleno do několika oblastí, jejichž prvky spolu úzce souvisí. Proto budu prvky popisovat právě po těchto skupinách.



Obrázek 1

- **SQL konzole (1) a příslušející tlačítka (2 a 3)**

Funkčnost SQL konzole zajišťuje jediná metoda, která reaguje na stisk tlačítka **2** i **3**. Třídy **OdbcConnection** a **OleDbConnection** podporují pouze odesílání jednotlivých příkazů. Jeden příkaz však může být v konzoli z důvodu přehlednosti napsán na více řádcích. Cílem této metody je analyzovat zapsané příkazy a text SQL konzole upravit tak, aby každý příkaz byl na samostatné řádce. Toho se docílí kontrolou uzávorkování, kdy je konec příkazu určen neuzávorkovaným středníkem. Pokud bylo stisknuto tlačítko **3**, je odeslán pouze příkaz na řádce s kurzorem, pokud tlačítko **2**, odešlou se postupně všechny příkazy, které SQL konzole obsahovala. Na závěr se ještě upravená verze SQL příkazů zkopíruje do skrytého textového pole, aby posloužila pro potřeby filtrování.

- **Výsledková tabulka (4) a příslušná tlačítka (5 a 6)**

Data získaná z databáze zobrazuje výsledková tabulka **4**. Třídění dat stisknutím nadpisu vybraného sloupce zajišťuje sama komponenta **DataGrid**. Stisknutím tlačítka **5** se vyvolá metoda **Update_Click**, která odešle upravená data zpět do databáze pomocí metody Update komponenty **OdbcDataAdapter** či **OleDbDataAdapter**. Stisknutím tlačítka **6** se spustí metoda **button2_click**, která zobrazí ve výsledkové tabulce (**4**) jen data zadaná kritérii vedle tlačítka filtruj (**6**). Nejprve načte z výsledkové tabulky právě označenou buňku s jejím nadpisem a poté sestaví potřebné SQL dotazy. Ve skrytém textovém poli jsou uloženy příkazy, které načetly filtrovaná data pro případ, že by uživatel změnil obsah konzole. Tyto příkazy metoda postupně bere a přidává jim na odpovídající místo klíčové slovo WHERE s patřičným omezením.

- **Object Browser (7) a příslušná tlačítka (8, 9, 10, 11)**

Object Browser zobrazuje podle právě zvolené záložky příslušné databázové objekty. Toho je docíleno sestavením dotazu do systémových tabulek a jeho odesláním do databáze. Výsledek se pak načte do Object Browseru pomocí

OdbcDataAdapter či **OleDbDataAdapter**. Object Browser se naplní v reakci na připojení databáze, změnu záložky, popřípadě stisknutí tlačítka **11**. Při stisknutí tlačítka **8** se vyvolá metoda **addname_Click**. Ta zjistí aktuálně označený řádek Object Browseru a název objektu, který je na řádku vypsán, se vloží na pozici kurzoru v SQL konzoli. Stisknutím tlačítka **10** se vyvolá průvodce přidáním či úpravou databázového objektu, který je právě zvolený. Jako parametr se konstruktoru předá název objektu. U objektů, pro které není průvodce podporován, je tlačítko neviditelné. Při stisknutí tlačítka **9** se vygeneruje skript, který vytvoří aktuálně vybraný databázový objekt a text skriptu se připojí na konec textu SQL konzole.

- **Informační lišta (13)**

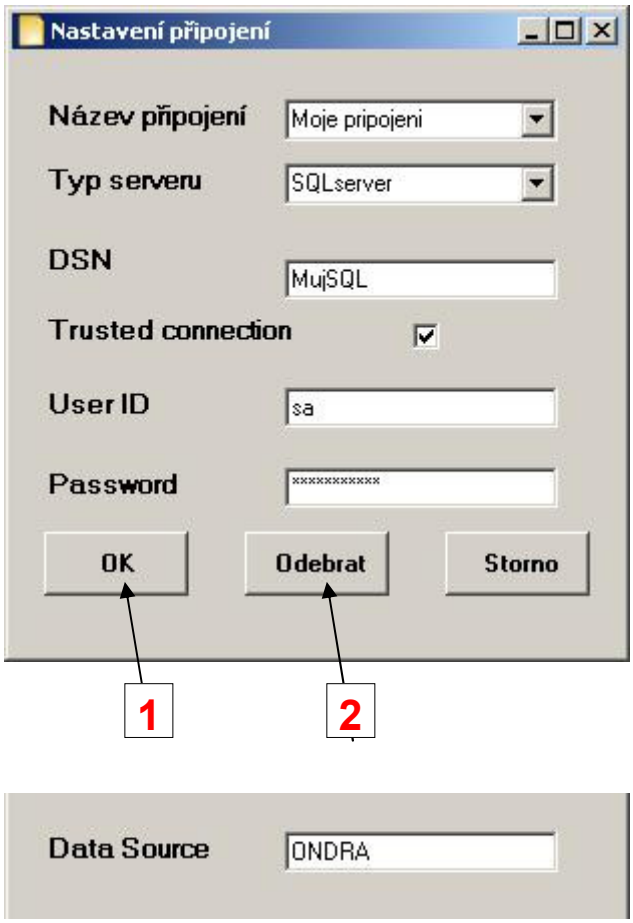
Informační lišta uživateli zobrazuje potvrzení a chyby, které v programu nastaly. Ukládá si všechny zprávy zobrazené od spuštění programu a obsluhují ji dvě metody – **error** a **message**. Metoda **message** slouží k vypisování informačních zpráv černou barvou a metoda **error** slouží k vypisování chybových hlášení červeně. Obě metody jsou určené k volání kdykoliv chce jiná metoda něco sdělit uživateli

- **Menu (14)**

V menu najde uživatel jak většinu funkcí, které jsou poskytovány tlačítky v okně, tak vedlejší funkce, jejichž umístění přímo do okna by bylo zbytečné. Netriviální funkci mají však jen tlačítka pro uložení a načtení výsledků ze souboru. Při ukládání se nejprve zjišťují nadpisy sloupců tabulky (**4**) a ukládají se do souboru oddělené tabulátorem a ukončené novou řádkou. Poté se postupně berou všechny řádky tabulky a hodnoty buněk oddělené tabulátory se taktéž ukládají do souboru. Načítání výsledků probíhá přesně inverzně k ukládání.

3.5.2 Okno Nastavení připojení

Okno Nastavení připojení poskytuje uživateli možnost, jak pohodlně nastavovat připojení k databázi.

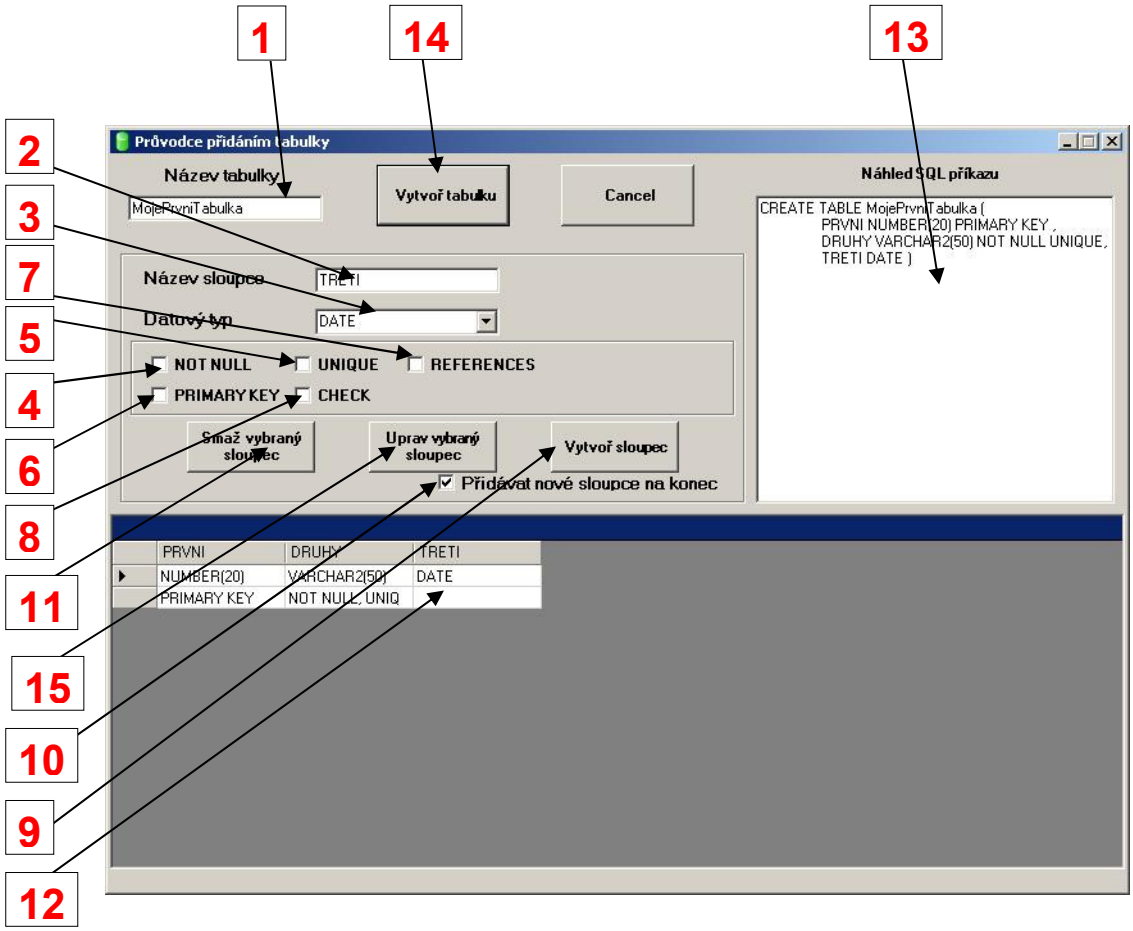


Obrázek 2

Při otevření okna se načtou prvky ze seznamu připojení **Pripoj** (struktura popsaná viz 3.3) a údaji se vyplní ovládací prvky okna. Uživatel pak může připojení editovat, případně odebírat ze seznamu uložených připojení. Tlačítkem OK se upravenými údaji aktualizuje seznam připojení **Pripoj** a do proměnné **Chosen** se uloží připojení, které bylo uživatelem zvoleno. Nakonec se seznam připojení pro jistotu uloží na disk.

3.5.3 Okno průvodce přidáním / úpravou tabulky

Průvodce úpravou tabulky se od průvodce přidáním tabulky liší jen drobně, a tak je popíši zároveň. Jednou z mála odlišností je inicializační fáze, ve které se do vnitřní reprezentace načte upravovaná tabulka. Ačkoliv to specifikace nepožadovala, ukázalo se jako přínosné, aby mohl uživatel SQL příkazy generované průvodcem před odesláním upravovat. Toto rozšíření výrazně zvyšuje možnosti využití zkušenějšími uživateli. Prvky budu stejně jako v případě hlavního okna popisovat po skupinách, které spolu souvisí.



Obrázek 3

- **Parametry nového sloupce (2 – 8, 10) a příslušná tlačítka (9, 11 a 15)**

Hlavní metodou této části okna je tlačítko **9**, které přidá sloupec se zadanými parametry na zvolené místo do seznamu sloupců **Tabulka** (struktura viz. 3.3).

Tlačítko **11** smaže sloupec, který si uživatel vybral v tabulce **12** ze seznamu sloupců **Tabulka**. Tlačítko **15** načte sloupec, který si uživatel v tabulce **12** vybral do ovládacích prvků **2** – **8** a taktéž ho smaže ze seznamu sloupců **Tabulka**. Při stisku tlačítka **15** navíc dočasně zmizí panel s integritními omezeními, jelikož jejich úprava průvodcem není podporována. Všechna tři tlačítka nakonec volají metody **ObnovMemo** a **ZapisDoTabulky**, které vnitřní změny prezentují uživateli.

- **Prezentace nové tabulky (12)**

Aktuálnost prezentované tabulky zajišťuje metoda **ZapisDoTabulky**. Je určena k tomu, aby byla volána z ostatních metod, kdykoliv se změní vnitřní reprezentace tabulky. Nejprve z vnitřní reprezentace (struktura viz. 3.3) načítá názvy sloupců nové tabulky, načež k nim do dvou řádků doplní jejich parametry a integritní omezení.

- **Náhled příkazu (13)**

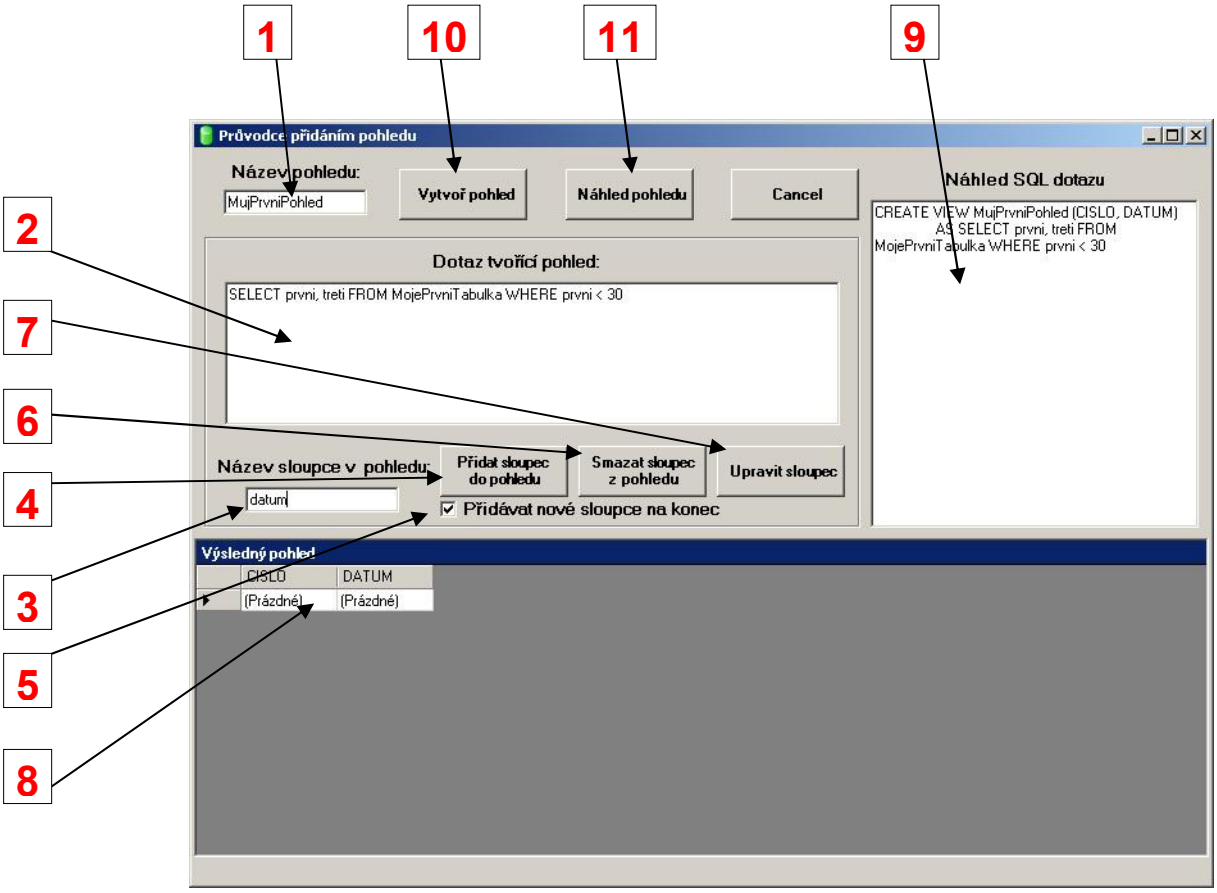
Aktuálnost okna, v němž je zobrazen náhled příkazu, který vytvoří tabulku se strukturou zobrazenou v **12**, zajišťuje metoda **ObnovMemo**. Je určena k tomu, aby byla volána z ostatních metod, kdykoliv se změní vnitřní reprezentace tabulky či název tabulky (**1**). Z vnitřní reprezentace načítá parametry sloupců tabulky a sestavuje k nim příslušné SQL příkazy, které buď vytvoří novou tabulku (v případě průvodce přidáním tabulky), nebo vybranou tabulku upraví (v případě průvodce úpravou tabulky).

- **Vytvoření/úprava tabulky (14)**

Vytvoření tabulky zajišťuje metoda **CreateTable_Click**. Pokud se vytváří nová tabulka, je příkaz pro přehlednost rozložen na více řádků. Jelikož však **OdbcCommad** i **OleDbCommand** nové řádky v příkazu nepodporují, jsou z příkazu nové řádky odstraněny. Poté se již příkazy odešlou do databáze a okno se zavře.

3.5.4 Okno průvodce přidáním / úpravou pohledu

Průvodce přidáním a úpravou pohledu se od předchozího okna (viz 3.5.3) liší jen ovládacími prvky, ale princip fungování je v postatě tentýž.



Obrázek 4

- **Parametry nového pohledu (2, 3 a 5) a příslušná tlačítka (4, 6 a 7)**

Hlavní metodou této části okna je tlačítko **4**, které přidá sloupec se zadaným názvem (**3**) na zvolené místo do seznamu sloupců (struktura viz. 3.3). Upravený seznam sloupců následně zapíše do tabulky (**8**) a zavolá metodu **ObnovMemo**, která aktualizuje Náhled SQL příkazu. Tlačítko **6** naopak smaže sloupec, který si uživatel

vybral v tabulce 8, z vnitřní struktury. Tyto změny, stejně jako metoda na vytvoření sloupce, zobrazí do vnější reprezentace Metoda skrývající se pod tlačítkem 7 se vnitřně příliš neliší od předchozího případu, jen před smazáním sloupce načte jeho název do kolonky (3), aby ho uživatel mohl změnit.

- **Prezentace nového pohledu (8)**

Aktuálnost tabulky se sloupci pohledu zajišťují samy metody, které mění vnitřní reprezentaci.

- **Náhled příkazu (13)**

Aktuálnost náhledu SQL příkazu, který vytvoří pohled se sloupci zobrazenými v tabulce 8 a dotazem zadaným v poli 2 zajišťuje metoda **ObnovMemo**. Je určena k tomu, aby byla volána z ostatních metod, kdykoliv se změní vnitřní reprezentace sloupců pohledu, název tabulky (1) či text SQL dotazu tvořícího pohled (2). Z vnitřní reprezentace načítá názvy sloupců pohledu a sestavuje postupně příslušný SQL příkaz, který vytvoří nový pohled.

- **Vytvoření/úprava pohledu (10) a náhled pohledu (11)**

Vytvoření pohledu i náhledu pohledu zajišťuje metoda **Preview_Click**. Pokud se pohled upravuje, je do databáze nejdříve zaslán příkaz na jeho vymazání. Pro přehlednost je text příkazu Náhledu SQL příkazu (9) rozložen na více řádků. Jelikož však **OdbcCommad** i **OleDbCommand** nové řádky v příkazu nepodporují, jsou z příkazu nové řádky odstraněny. Poté je příkaz odeslán do databáze, a pokud uživatel stisknul tlačítko „Vytvoř pohled“ 10, okno se zavře. Pokud chtěl jenom vidět náhled pohledu, provede se příkaz SELECT na nově vytvořený pohled, výsledek se zapíše do tabulky (8) a pohled se smaže.

3.6 Finální verze vs. specifikace

Finální verze se od specifikace liší jen velmi málo. Odchylky jsou způsobené tím, že jsem na začátku měl jen malý přehled o databázích, a tak byly některé požadavky ve specifikaci zbytečné, a naopak chyběly požadavky na prvky programu, jejichž absence by práci znepříjemňovala a narušovala by kompaktnost programu jako nástroje. Myslím si však, že tyto drobné odchylky nejsou na škodu.

Jako příklad uvádím jen velice jednoduché filtrování, které specifikace navrhovala, ale při praktickém zkoušení aplikace vyšlo brzy najevo, že pro většinu požadavků na filtrování není dostačující a že je potřeba jej rozšířit.

Jako opačný příklad uvádím navrhovaného průvodce na vytváření procedur, který se po proniknutí do syntaxe SQL ukázal být zbytečným, protože by uživateli práci nijak neulehčil, vzhledem k tomu že tělo procedury – tedy většinu práce – by si musel uživatel stejně napsat sám.

Kapitola 4 – Dokumentace

4.1 Specifikace

Specifikace je dokument, který předchází vzniku softwarového díla a měl by odrážet požadavky zadavatele zjištěné v průběhu analýzy a návrhu aplikace. V závěru vývoje pak slouží jako podklad k hodnocení, nakolik se povedlo splnit vytyčené cíle.

Specifikace tvoří v podstatě rozhraní mezi zadavatelem a zpracovatelem softwarového díla a slouží k jasnému stanovení požadavků na program. A právě vzhledem k tomu, že specifikace musí sloužit dvěma oddílným světům (laik vs. programátor), je třeba vyvážit dvě protichůdné vlastnosti: Specifikace musí být napsaná bez zbytečných technických obratů a odborných termínů, aby byla pochopitelná i pro zadavatele - laika, ale zároveň dostatečně konkrétní a popisná, aby neumožňovala různé výklady ze strany programátora.

Má specifikace je proto psaná co nejčtivěji, ovšem zároveň poskytuje zcela konkrétní seznam funkcí a vlastností, které má výsledný program mít. Specifikace programu DBManager je součástí této práce a najdeme ji v kapitole Přílohy.

4.2 Uživatelská příručka

Uživatelská příručka je dokument, který popisuje program z uživatelského hlediska. Základním požadavkem je, aby jí porozuměl i člověk naprosto neznalý řešené problematiky. Je také vhodné, aby čtenáře nezatěžovala odbornými termíny (až na nezbytné minimum) a byla co nejpochopitelnější.

Na začátku uživatelské dokumentace čtenáře seznamuji s charakteristikou programu, aby si udělal přibližnou představu. Dále ho krok za krokem provedu instalací a spuštěním programu, čímž vlastně do uživatelské příručky zahrnuji, někdy samostatně uváděnou, instalační příručku. Vzhledem k tomu, že program je postaven na reakcích na události, není možné ho popisovat lineárně. Pro přehlednost jsem tedy zvolil formu popisu jednotlivých prvků, jejich funkce a případné interakce s ostatními prvky.

Dokument je samozřejmě doplněn obrazovkami programu s popisem, které ho dělají přehlednějším a názornějším. Uživatelskou příručku tak lze číst i bez spuštěné aplikace či dokonce počítače. Dále je příručka doplněna obvyklými chybami, se kterými se může uživatel setkat, jejich možnými příčinami a nástinem jejich řešení. Příručku uzavírá minimální konfigurace počítače a softwarové prerekvizity. Uživatelská příručka programu DBManager je součástí této práce a najdeme ji v kapitole Přílohy.

4.3 Programátorská příručka

Programátorská příručka popisuje strukturu programu DBManager, jeho netriviální datové struktury a metody, které zajišťují jeho funkčnost. Je psána pro programátora, který by se případně snažil můj program v budoucnosti modifikovat či rozšiřovat. Nepopisuje běžné programovací obraty (ty důležitější jsou okomentované přímo v programu), ale místo toho se snaží vystihnout smysl jednotlivých celků programu.

Pro přehlednost jsem zvolil podobný systém jako v uživatelské dokumentaci, kde je popis programu členěn do kapitol pro každé okno. V úvodu je obrázek okna s popisky, na které se pak v textu odvolávám, aby byl popis názorný a nemohlo dojít k nedorozumění. V každé kapitole jsou dále popisovány samostatně metody, které zajišťují funkčnost jednotlivých prvků programu. Programátorská příručka programu DBManager je součástí této práce a najdeme ji v kapitole Přílohy.

Kapitola 5 – Závěr

5.1 Získávání informací

Na začátku psaní programu jsem o SQL a platformě .NET věděl jen velmi málo, a tak jsem se vlastně do psaní program pouštěl jako na cestu do neznáma. Určitě nebyla snadnou a často jsem narážel na nedostatečnou znalost syntaxe SQL či platformy .NET. Znalosti o jazyku SQL jsem čerpal hlavně z Kapesního průvodce SQL [3] a webové stránky [4], a také samozřejmě z různých dalších webových stránek pomocí nepostradatelného Google [1]. Znalosti o .NETu jsem pak plně čerpal z MSDN [2]. Pokud jsem navíc potřeboval nadhled a srovnání různých technologií, bohatě mi posloužila encyklopedie Wikipedia [5].

5.2 Zhodnocení

Pravděpodobně díky tomu, že toto je moje první větší softwarové dílo, jsem se při samotném programování často pozastavoval a dlouho hledal banální, ale o to důkladněji schované chyby. Avšak s postupem implementace se díky těmto obtížím také zvyšoval můj cit pro hledání chyb, což pro mě bylo ve výsledku přínosem.

Je docela zajímavé, že s tím jak postupovala implementace a program získával na funkčnosti jsem jej začal používat jako pomůcku k jeho vlastnímu dalšímu vývoji. Kupříkladu jakmile jsem dokončil SQL konzoli s výstupem do tabulky, používal jsem ji k prohlížení systémových tabulek, abych správně zformuloval dotazy s nimi spojené při programování Object Browseru. S jeho dokončením se práce ještě zpříjemnila a od té chvíle (tedy při programování průvodců) jsem již v podstatě jiný nástroj nepoužíval.

Pro tento zajímavý „rekurzivní“ jev, kdy jsem program použil k jeho vlastnímu vývoji, nabízí zajímavá paralela s relačními databázemi, se kterými pracuje, kde je struktura databáze uložená v databázi samotné.

Jako hlavní důvody, proč jsem používal můj program a ne jiné nástroje (SQL tools, Toad, atd.), vidím výhodu možnosti připojit se jak na MS SQL server, tak Oracle a

uživatelskou přívětivost. To byly také cíle, které jsem po analýze podobných dostupných programů (viz 2.3) stanovil, a tímto se potvrdila správnost zmíněné analýzy. Stejně tak se tím potvrdilo, že nástroj opravdu pokrývá potřeby cílové skupiny určené v úvodu práce (viz 1.1).

Programováním tohoto projektu jsem se obohatil o množství zkušeností v oblasti databází, objektového programování pod Windows i v oblasti samotného plánování, analýzy a implementace software.

5.3 Možnosti dalšího vývoje

V průběhu vývoje aplikace se objevilo množství dodatečných funkcí a nápadů, které však nebyly součástí specifikace (viz 2.4). Některé z nich byly pro svou nepostradatelnost dodatečně implementovány. Většinu z nich však z časových důvodů nebylo možné realizovat, a tak je zde uvádím jako inspiraci k dalšímu rozšiřování a vylepšování programu:

- **Syntax Highlighting**

Velice příjemná vlastnost, kdy se klíčová slova automaticky převádějí na UpperCase. Této vlastnosti jsem si všiml u programu SQL tools a výrazně zpřehledňuje práci.

- **Rozšíření možností konzole**

Například přidání tlačítek „Undo“ a „Redo“, jaká jsou běžná v textových editorech. Rozšíření není vůbec složité, jelikož podporu pro tato tlačítka zajišťuje přímo komponenta tvořící SQL konzoli. Také by bylo příjemné přidat tlačítka pro záložky (jak je známe například z Microsoft Visual Studia), které by zrychlily orientaci ve větších SQL skriptech.

- **Rozšíření možností připojení**

Program zatím podporuje jen dvě nejrozšířenější databáze: Oracle a MS SQL server. Je však bezpočet dalších, také dost rozšířených databází, pro které by se dala přidat podpora: MySQL, Informix, SyBase… Vzhledem k tomu, že program používá ODBC,

které podporuje dnes již drtivá většina databází, nebyl by problém se na databázi připojit. Jen by se musely upravit ty části programu, které může mít databáze specifické:

- Připojovací údaje (každá databáze mívá svojí syntaxi připojovacího řetězce)
- Datové typy – ačkoliv existují normy SQL, ne vždy jsou dodržovány
- Veškeré operace se systémovými tabulkami, tedy Object Browser a průvodci na úpravu tabulky/pohledu (konkrétně jejich metoda load, kde se načítá ze systémových tabulek)

- **Rozšíření funkcí pro práci s databází**
  - Statistiky serveru
  - Tlačítko na „explain plan“
  - Hledání databázových objektů

- **Načítání dat po částech**

U hodně velkých tabulek (popřípadě při pomalém připojení) může být prodleva, než se tabulka načte, relativně dlouhá. Řešením by bylo načítat tabulku po částech a další část načíst, až když ji chce uživatel opravdu vidět. Ovšem má to i své nevýhody – tuto vlastnost má právě aplikace SQL tools (viz 2.3) a není moc příjemné, když projíždění tabulkou není plynulé a co chvíli se zastaví, aby se načetly další data.

- **Výběr oddělovače**

Program ukládá data z tabulky s výsledky oddělené tabulátorem. Uživatel by si však mohl přát zvolit jiný oddělovač (věděl by kupříkladu, že se v ukládané tabulce tabulátory vyskytují, ale jiný ASCII znak ne)

- **Rozšířit průvodce**

Již vytvoření průvodci by se jistě dali rozšířit o další detaily syntaxe. Také by se dalo uvažovat o průvodcích pro další, méně používané, databázové objekty.

Reference

[1] Google
http://www.google.com/

[2] Microsoft Corp. The MSDN Library
http://msdn.microsoft.com/library

[3] Šimůnek M.: SQL Kompletní kapesní průvodce, GRADA Publishing, spol. s r.o., Havlíčkův Brod, 1999.

[4] Tech on the Net
http://www.techonthenet.com/

[5] Wikipedia
http://www.wikipedia.org/

[6] Microsoft Corp. ODBC
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/dasdkodbcoverview.asp

[7] Microsoft Corp. OLE DB
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/dasdkoledboverview.asp

[8] Microsoft Corp. ADO
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/dasdkadooverview.asp

[9] SQL-92 standart
http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

[10] Oracle PL/SQL Technology Center
http://www.oracle.com/technology/tech/pl_sql/index.html

[11] Microsoft Corp. Transact-SQL reference
http://msdn2.microsoft.com/en-us/library/ms189826.aspx

[12] T&T Solutions inc. DB Commander
http://www.dbcommander.com/

[13] Cardett Associates Ltd. Advanced Query Tool
http://www.querytool.com/

[14] Quest Software Inc. Toad
http://www.quest.com/

[15] Aleksey Kochetov SQL Tools
http://www.sqltools.net/
[16] Microsoft Corp. Visual studio .NET 2003
http://msdn.microsoft.com/virtuallabs/vstudio/
[17] Microsoft Corp. MSDN Academic Aliance Software Distribution (ELMS)
http://msdn.microsoft.com/academic/program/eacademy/default.aspx
[18] Microsoft Corp. .NET Homepage
http://www.microsoft.com/net/default.mspx

Přílohy