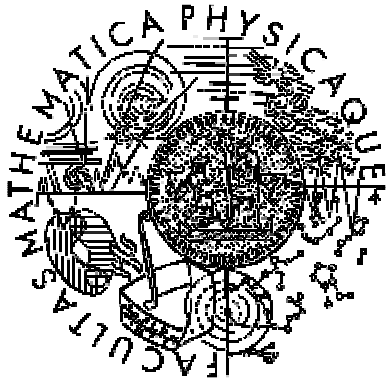
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Viliam Sabol

Demonstrační aplikace vyhodnocování dotazu v relačním kalkulu

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Tomáš Skopal, Ph.D.

Studijní program: Informatika, programování

2007

Ďakujem pánovi RNDr. Tomášovi Skopalovi Ph.D. za odborné vedenie, za ochotu a čas, ktorý mi venoval počas písania tejto bakalárskej práce.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných zdrojov. Súhlasím so zapožičiavaním práce.

V Prahe dňa 3.7.2007 Viliam Sabol

Obsah

Názov práce: Demonstrační aplikace vyhodnocování dotazu v relačním kalkulu
Autor: Viliam Sabol
Katedra: Katedra softwarového inženýrství
Vedúci bakalárskej práce: RNDr. Tomáš Skopal, Ph.D.
E-mail vedoúceho: Tomas.Skopal@mff.cuni.cz
Abstrakt:
Predmetom práce je implementácia výukovej aplikácie, ktorá vyhodnocuje databázový dotaz v n-ticovom relačnom kalkule. Dôraz sa kladie na vizualizáciu štruktúry dotazu, krokovanie a vizualizáciu vyhodnocovania dotazu. Aplikácia je určená predovšetkým na vysvetlenie procesu vyhodnotenia dotazu v n-ticovom relačnom kalkule nad jednoduchou, užívateľom vytvorenou databázou. Výstupom aplikácie je vykreslenie stromovej štruktúry dotazu, ako aj výstup vyhovujúcich dát z databázy. Aplikácia umožňuje pozastaviť a krokovať proces vyhodnocovania, sledovať hodnoty premenných použitých v dotaze, a výsledky operácií nad týmito hodnotami.

Kľúčové slová: n-ticový relačný kalkul, vyhodnotenie dotazu, vizualizácia dotazu

Title: A demonstrational application for relational calculus query evaluation
Author: Viliam Sabol
Department: Department of Software Engineering
Supervisor: RNDr. Tomáš Skopal, Ph.D.
Supervisor's e-mail address: Tomas.Skopal@mff.cuni.cz
Abstract:
The task of this thesis is implementation of an application for educational purposes, which will be used for evaluating queries in tuple relational calculus. It is aimed at visualization of query structure and evaluation process. The application is especially intended for understanding evaluation process of queries in tuple relational calculus on simple user defined database. The output of the application is a tree structure of query, and requested data from database. The application allows user to trace the evaluation process and watch values of variables used in query and results of operations in query tree.

Keywords: tuple relational calculus, query evaluation, query visualization

1 Úvod

V súčasnosti sa použitie databáz dostáva čoraz viac do každodenného života. Od databázy dopravných spojení až po databázu hudby v osobnom prehrávači, vo všetkých týchto oblastiach sa uplatňujú databázové systémy. Je potrebné spravovať stále viac informácií a nároky na rýchlosť vypracovania odpovede sú tiež čoraz vyššie. To vedie k neustálemu vylepšovaniu vnútorných štruktúr databáz a je len na užívateľovi ako sformuluje požiadavku na dáta, ktoré z databázy potrebuje získať. Pre užívateľa je dôležité poznať vlastnosti dotazovacieho jazyka, aby dokázal efektívne získať a využiť informácie z databázy.

Táto práca sa zaoberá n-ticovým relačným kalkulom, jedným z dotazovacích jazykov určených pre relačné databázy. Relačný kalkul predstavil Edgar F. Codd v rámci relačného modelu dát ako neprocedurálny dotazovací jazyk. Existujú dva varianty tohto jazyka. N-ticový relačný kalkul pracuje s reláciou ako s množinou n-tíc, zatiaľ čo doménový relačný kalkul pracuje s hodnotami jednotlivých n-tíc.

Cieľom tejto práce bolo vytvoriť vhodné prostredie pre prezentáciu vyhodnotenia databázového dotazu zapísaného v n-ticovom relačnom kalkule tak, aby užívateľ pochopil princíp vyhodnocovania dotazu nad databázou. Výsledkom práce je aplikácia TRC, ktorej názov je odvodený od anglického názvu n-ticového relačného kalkulu, Tuple Relational Calculus. Aplikácia TRC je určená predovšetkým na výukové účely a umožňuje vyhodnocovanie dotazu a krokovanie procesu vyhodnotenia. Užívateľské rozhranie aplikácie umožňuje jednoduché vytvorenie databázy, ako aj ukladanie a načítanie projektov tvorených súborom dotazov, databázových tabuliek a poznámok. Značenie syntaxe dotazovacieho jazyka je podobné ako značenie používané pri výuke predmetu Databázové systémy na MFF UK.

Práca obsahuje popis vlastností n-ticového relačného kalkulu a porovnanie s ďalšími dotazovacími jazykmi. V nasledujúcej kapitole sú uvedené požiadavky ktoré má aplikácia spĺňať. Štvrtá kapitola obsahuje užívateľskú dokumentáciu a podrobný popis užívateľského prostredia aplikácie. Piata kapitola sa zaoberá problémami pri vytváraní grafického užívateľského rozhrania a vyhodnocovacieho mechanizmu z programátorského hľadiska. Záverečná časť práce obsahuje porovnanie s inými výukovými aplikáciami podobného typu a vzorové ukážky výstupu aplikácie.

2 Teória

2.1 Relačný model dát

N-ticový relačný kalkul je nástrojom na získavanie dát z relačných databáz. Pred samotnou definícou vlastností tohto dotazovacieho nástroja je nutné definovať vlastnosti relačnej databázy.

S návrhom relačného modelu dát prišiel ako prvý E. F. Codd. V článku [1] navrhol okrem samotného relačného modelu aj nástroje na prácu s údajmi a to relačný kalkul a relačnú algebru.

Základným pojmom pre relačný model dát je relácia. Rozlišujeme pojmy: [2]

- schéma relácie – určuje meno relácie, množinu atribútov, a pre každý atribút definuje meno a doménu atribútu. Doména atribútu je špecifikovaná svojím menom a obsahuje presne definovanú množinu hodnôt, ktoré môže atribút s touto doménou nadobúdať.

- inštancia relácie – je tvorená množinou záznamov, alebo n-tíc, ktoré odpovedajú schéme relácie. Znamená to, že každá n-tica obsahuje rovnaký počet atribútov rovnakej domény ako určuje pre každý atribút schéma. Namiesto pojmu inštancia relácie sa často používa len pojem relácia.

V praxi si môžeme predstaviť inštanciu relácie ako tabuľku naplnenú údajmi, a schému relácie ako hlavičku tejto tabuľky. Riadky tabuľky predstavujú n-tice ktoré majú v rámci jednej tabuľky rovnaký počet atribútov. Vlastnosťou relačného modelu dát je tzv. 1. normálna forma relácie [3], a znamená, že hodnoty riadkov tabuľky sú atomické. V relácii nezáleží na poradí n-tíc, teda na poradí riadkov v tabuľke, ale dôležité je, že tabuľka neobsahuje žiadne dve rovnaké n-tice. Ďalším typom obmedzenia v relačnom modeli je doménové obmedzenie, ktoré pre každý atribút relácie určí jeho doménu. Takto vznikne väzba medzi hodnotou v každom stĺpci riadku a typom ktorý určuje množinu hodnôt ktoré sa v tomto stĺpci tabuľky môžu vyskytovať. Pre relačnú schému

$$R(a_1{:}D_1, a_2{:}D_2, \dots, a_n{:}D_n)$$

kde a_i je atribút, D_i doména a Dom_i je množina hodnôt domény D_i, $1 \leq i \leq n$. Potom n-tica ktorá spĺňa doménové obmedzenie je definovaná takto:

$$\{ a_1{:}d_1, a_2{:}d_2, \dots, a_n{:}d_n \}, \text{ kde } d_i \in Dom_i, 1 \leq i \leq n.$$

Pre reláciu ešte definujeme pojmy arita, teda počet atribútov (stĺpcov), a kardinalita, alebo počet n-tíc (riadkov). Relačnú databázu definujeme ako množinu relácií

s jednoznačnými menami, a ako schému relačnej databázy označíme súbor relačných schém jednotlivých relácií. Dôležitým pojmom je aktuálna doména atribútu, ktorá predstavuje množinu hodnôt, ktoré sa pre daný atribút vyskytujú v databáze. Ďalší druh obmedzení, tzv. integritné obmedzenia, ktoré zaisťujú správnosť dát vkladaných do databázy, tvoria osobitnú kapitolu v rámci relačného modelu. Vzhľadom k tomu že integritné obmedzenia sa priamo netýkajú spôsobu vyhodnocovania dotazov, nebudem sa im v práci hlbšie venovať. Príkladom relácie je nasledujúca relácia *FILM* a *HEREC*. Schémy týchto relácií sú podľa definície:

FILM(MENO_FILMU: string, ROK: integer, MENO_HERCA: string)
HEREC(MENO_HERCA: string, NARODNOST: string)

FILM

MENO_FILMU	ROK	MENO_HERCA
Rocky	2002	Stallone
Silny kafe	2004	Labus
Okenka	2001	Labus
Rumburak	1984	Bedrna
The Kid	2002	Perry
Friends	2004	Perry
D-Tox	2002	Stallone
Letiste	2006	Kramar
Smrt nacerno	1977	Labus
Smrt nacerno	1977	Bedrna

HEREC

MENO_HERCA	NARODNOST
Stallone	USA
Bedrna	CZ
Labus	CZ
Perry	USA
Kramar	SK
Landa	CZ
Craig	USA
Stiller	USA
Streep	USA
Jackman	USA

Obr. 2.1 Príklad relácie – relácia FILM a HEREC

2. 2 N-ticový relačný kalkul (NRK)

Relačný kalkul je jeden z dotazovacích jazykov nad relačnou databázou, teda nástroj, ktorý umožňuje zadávať dotazy pre databázu. Pri vyhodnotení dotazu sa použije relácia z databázy, a výstupom každého dotazu je tiež relácia. Relačný kalkul je neprocedurálny dotazovací jazyk, čo znamená, že dotaz sa vytvorí popisom vlastností dát, ktoré z databázy chceme dostať. Dotaz však nešpecifikuje spôsob akým tieto údaje z databázy vybrať. Relačný kalkul silne ovplyvnil aj vývoj komerčných dotazovacích jazykov ako je v súčasnosti populárny SQL, a existujú dva varianty relačného kalkulu: doménový a n-ticový rel. kalkul, ktorým sa táto práca zaoberá. Už z názvu vyplýva, že NRK (n-ticový rel.kalkul) berie reláciu ako množinu n-tíc. NRK používa v dotazoch premenné, ktoré nadobúdajú hodnoty n-tíc relácie, teda n-tíc danej schémy relácie. Pre každú premennú ktorá sa v dotaze použije by malo byť určené meno relácie. Množina n-tíc tejto relácie je potom použitá ako množina hodnôt premennej.

Dotaz zapísaný v NRK má tento tvar: [2]

$$\{ v \mid f(v) \}$$

kde *v* je premenná a *f(v)* je formula NRK obsahujúca premennú *v*. Výsledkom tohoto dotazu je množina všetkých n-tíc *t*, pre ktoré sa po dosadení $v = t$ vyhodnotí formula *f(v)* ako pravdivá formula. Vyhodnotenie formuly úzko súvisí s predikátovou logikou prvého rádu, ktorá používa rovnaké logické operácie ako sú použité v jazyku NRK. Pre formálnu definíciu syntaxe NRK sa používajú pojmy atóm (atomická formula) a formula. Najprv definujeme atomickú formulu. Nech *u*, *v* sú premenné, *a* atribút premennej *u*, *b* atribút premennej *v*, a *k* nech je konštanta. Ďalej označme *Rel* identifikátor relácie a ***op*** ako jeden operátor z množiny $\{ <, >, =, \neq, \leq, \geq \}$. Potom ***atomická formula*** má jeden z tvarov:

1. *Rel(v)*
2. *u.a* ***op*** *v.b*
3. *u.a* ***op*** *k*, *k* ***op*** *u.a*

Príkladom atomických formúl pri použití relácie *FILM* z *obr. 2. 1* sú tieto formuly:

1. *FILM(v), HEREC(u)*
2. *v.MENO_HERCA = u.MENO_HERCA*
3. *v.ROK < 2002*

Nech *f* a *g* sú formuly NRK, a *f(v)* je formula, ktorá obsahuje premennú *v*. ***Formulu NRK*** definujeme rekurzívne a pomocou atomických formulí takto:

1. každá atomická formula je formulou NRK
2. formuly $\neg f$, $f \wedge g$, $f \vee g$, $f \Rightarrow g$, $f \Leftrightarrow g$ sú formulami NRK. Kde symbol $\neg$ je negácia, $\wedge$ konjunkcia, $\vee$ disjunkcia, $\Rightarrow$ implikácia a symbol $\Leftrightarrow$ označuje logickú ekvivalenciu.
3. formuly $\exists v\ (f(v))$, $\forall v\ (f(v))$ sú formulami NRK.

Symbol $\forall$ je všeobecný, a $\exists$ existenčný kvantifikátor. Ak označíme *Q* ako kvantifikátor, potom formula *Q v (f(v))* znamená, že kvantifikátor *Q* viaže výskyt premennej *v* v podformule *f(v)*. Teda premenná *v* sa v takomto prípade nazýva viazaná. Premenná je vo formule voľná, ak nieje v tejto formule viazaná kvantifikátorom. S pomocou vyššie uvedených definícií formúl môžeme formálne zadefinovať aj dotaz NRK, ktorý má tvar [2] *{ v | f(v) }*, kde *v* je jediná voľná premenná v rámci formule *f*.

Vyhodnotenie dotazu NRK prebieha vzhľadom k relačnej databáze. Výstupom dotazu je množina n-tíc, pre ktoré sa po dosadení do formuly dotazu za hlavnú premennú, vyhodnotí táto formula ako pravdivá. Atomická formula typu *Rel(v)* sa vyhodnotí ako pravdivá, ak sa do premennej *v* dosadí n-tica ktorá patrí relácii *Rel*. Pre atóm typu *u.a* ***op*** *v.b*, kde ***op*** označuje porovnávajúci operátor, sa formula vyhodnotí ako pravdivá v prípade, že podľa schémy relácie majú atribúty premenných *u* a *v* rovnaký typ, a výsledok porovnania hodnôt atribútov je pravdivý. Atóm v ktorom sa porovnáva

hodnota atribútu premennej s konštantou musí tiež spĺňať požiadavok na rovnosť typov atribútu a konštanty, a vyhodnotením je takisto výsledok porovnania hodnoty atribútu a konštanty. Pri vyhodnocovaní formúl NRK sa postupuje podľa pravidiel predikátovej logiky 1. rádu, teda formula $\neg f$ platí len v prípade že formula *f* neplatí. Formula $f \wedge g$ je pravdivá len v prípade, že *f aj g* sú pravdivé. Formula $f \vee g$ platí, ak platí aspoň jedna z formúl *f* alebo *g*. Pre formulu v tvare implikácie $f \Rightarrow g$, je vyhodnotenie nepravdivé len v prípade, že *f* platí a *g* neplatí. Formuly v tvare ekvivalencie $f \Leftrightarrow g$ sa vyhodnotia ako pravdivé v prípade, že pravdivosť obidvoch formúl *f* aj *g* je rovnaká. Formula tvaru $\exists v$ *(f(v))* sa vyhodnotí ako pravdivá práve vtedy, ak existuje aspoň jedna n-tica v relácii, ktorá po dosadení do premennej *v* spôsobí, že sa formula *f(v)* vyhodnotí ako pravdivá. Formula $\forall v$ *(f(v))*, so všeobecným kvantifikátorom, sa vyhodnotí ako pravdivá, ak pre všetky n-tice relácie, ktoré je možné dosadiť za premennú *v*, sa formula *f(v)* vyhodnotí ako pravdivá. Inými slovami, ak existuje také priradenie n-tíc k voľným premenným vo formule *f(v)*, pre ktoré je táto formula pravdivá, bez ohľadu na to, aká n-tica sa dosadí za premennú *v*.

Operácia projekcia, ktorá sa používa v atomických formulách na výber atribútu z n-tice, sa v dotazoch používa (v odlišnom značení) aj na projekciu hlavnej premennej dotazu na jeden alebo viac atribútov. Príkladom NRK dotazu nad rel. databázou tvorenou reláciami na obr. 2.1 je dotaz ktorý vyberá všetky filmy v ktorých hrajú českí herci.

{ f[MENO_FILMU] | FILM(f) $\wedge$ *∃h(HEREC(h)* $\wedge$ *f.MENO_HERCA = h.MENO_HERCA* $\wedge$ *h.NARODNOST = 'CZ') }*

Výstupom dotazu je táto relácia:

RESULT

	f.MENO_FILMU
	Silny kafe
	Okenka
	Rumburak
	Smrt nacerno

2.3 Bezpečné formuly NRK

Niektoré dotazy NRK vedú k nekonečným výsledným reláciám napriek tomu že sú syntakticky správne. Takýmto príkladom je dotaz *{ h |* $\neg$ *(HEREC(h)) }*. Výstupom tohto dotazu by mali byť všetky n-tice ktoré sa nenachádzajú v danej inštancii relácie *HEREC*. Množina týchto n-tíc je nekonečná a dotazy tohoto typu niesú bezpečné. Aby sa predišlo takýmto problémom, môžeme vyhodnocovať dotaz len nad aktuálnou doménou každého atribútu [3]. Takémuto vyhodnocovaniu dotazov NRK hovoríme vyhodnotenie

s obmedzenou interpretáciou. Vhodné je obmedziť množinu dotazov NRK tak aby obsahovala len bezpečné formule. Označme *I* množinu relácií ktoré sa vyskytujú v dotaze *D*. *Dom(I, D)* je množina všetkých konštánt a hodnôt ktoré sa nachádzajú v množine relácií *I* alebo v samotnom dotaze *D*. Predpokladáme len konečné relácie a preto aj *Dom(I, D)* je konečná. Podľa [2] sú bezpečné dotazy také, ktorých výsledok tvoria len n-tice, ktorých hodnoty sa vyskytujú v *Dom(I, D)*, a zároveň platí, že pri vyhodnotení sa na zostavenie výslednej relácie použijú tiež len hodnoty z *Dom(I, D)*. Presná definícia bezpečných formúl [2]:

1. Pre každú množinu relácií *I*, výsledok dotazu *D* obsahuje len hodnoty ktoré obsahuje *Dom(I, D)*.
2. Pre každú podformulu dotazu *D* tvaru $\exists v\ (\ f(v))$ platí: Ak pre n-ticu *t*, dosadenú do formuly (za premennú *v)* daná formula platí, potom n-tica *t* obsahuje len hodnoty z *Dom(I, D)*.
3. Pre každú podformulu dotazu *D* tvaru $\forall v\ (\ f(v))$ platí: Ak n-tica *t*, ktorú dosadíme do formuly (za premennú *v)* obsahuje hodnotu ktorá nieje v *Dom(I, D)*, potom je táto formula pre $v = t$ pravdivá.

Aj keď táto definícia vymedzuje množinu bezpečných formúl v rámci NRK, nehovorí ako možno otestovať či daná formula je bezpečná, alebo nieje.

Vyhodnotenie dotazu, ktorý nespĺňa podmineky bezpečnosti, nespôsobí žiadne problémy, pretože sa dotaz vyhodnocuje podľa aktuálnej domény. Preto kontrola na bezpečnosť formúl dotazu prebieha v aplikácii TRC až po jeho vyhodnotení, a oznámi či je formula dotazu bezpečná. Bezpečný dotaz v aplikácii TRC spĺňa nasledujúce podmienky [6]:

1. Pre každú operáciu $\vee$ (disjunkcia) platí, že pravá aj ľavá podformula operácie obsahuje rovnaké voľné premenné.
2. Pre každú voľnú *v* premennú v maximálnej konjunkcii $f_1 \wedge f_2 \wedge \ldots f_n$ platí, že je *ohraničená*, teda platí pre ňu aspoň jedna z podmienok:
    a. premenná je voľná v f_i, ktorá nieje negáciou ani porovnaním.
    b. existuje f_i tvaru $v = k$, kde *k* je konštanta.
    c. existuje f_i tvaru $v = u$, kde *u* je ohraničená.
3. Negácia sa vyskytuje len v konjunkcii $f_1 \wedge f_2 \wedge \ldots f_n$.

Ďalším variantom relačného kalkulu je doménový relačný kalkul, ktorý sa líši od NRK tým, že nepracuje s n-ticami, ale s ich jednotlivými hodnotami. DRK používa tzv. doménové premenné, za ktoré sa dosadzujú hodnoty jednotlivých atribútov relácií. V rámci relačného modelu dát sa sila dotazovacieho jazyka často hodnotí porovnaním s relačnou algebrou ktorá je silným dotazovacím nástrojom. Narozdiel od relačného kalkulu je pre relačnú algebru vstupom aj výstupom pre každú z operácii relácia. Pomocou bezpečných formúl relačného kalkulu dokážeme vyjadriť všetky dotazy, ktoré sa dajú formulovať v jazyku relačnej algebry. Dotazovacím jazykom ktoré spĺňajú túto požiadavku sa hovorí relačne kompletné jazyky.

2.4 Vyhodnocovanie dotazu v Aplikácii TRC

Vyhodnocovanie dotazov prebieha v aplikácii TRC podľa aktuálnej domény. Aplikácia teda správne vyhodnotí len bezpečné dotazy. Vyhodnotenie prebieha v dvoch fázach. Najprv sa zo vstupného dotazu typu *{ v | f(v) }* postaví stom formuly *f(v)*, ktorého uzly sú tvorené jednotlivými operáciami a ich operandami. Ďalšou fázou je samotné vyhodnocovanie pomocou vytvoreného stromu dotazu. Stavba stromu začína parsovaním textového vstupu ktorý sa rozdeľuje na jednotlivé časti, ktoré predstavujú operandy logických a ďalších operátorov. Okrem štandarndých operácií ktoré zahŕňa definícia jazyka NRK, je možné v aplikácii TRC použiť aj logickú implikáciu, ekvivalenciu, ako aj jednoduché matematické operácie +, -, *, /, a skrátený zápis vnorených kvantifikačných formúl. Parsovanie a stavba stromu sa riadi nasledujúcou gramatikou:

$$
\begin{aligned}
\langle \textit{Dotaz} \rangle &\rightarrow \{ \langle \textit{Premenná} \rangle \mid \langle \textit{Formula} \rangle \} \\
\langle \textit{Premenná} \rangle &\rightarrow \textit{identifikátor} \\
&\mid \textit{identifikátor}, \langle \textit{Premenná} \rangle \\
&\mid \textit{identifikátor}[\langle \textit{Identifikátory} \rangle] \\
&\mid \textit{identifikátor}[\langle \textit{Identifikátory} \rangle], \langle \textit{Premenná} \rangle \\
\langle \textit{Identifikátory} \rangle &\rightarrow \textit{identifikátor} \\
&\mid \textit{identifikátor}, \langle \textit{Identifikátory} \rangle \\
\langle \textit{Formula} \rangle &\rightarrow \langle \textit{EqFormula} \rangle \\
&\mid \langle \textit{EqFormula} \rangle \Rightarrow \langle \textit{Formula} \rangle \\
\langle \textit{EqFormula} \rangle &\rightarrow \langle \textit{AndFormula} \rangle \\
&\mid \langle \textit{AndFormula} \rangle \Leftrightarrow \langle \textit{EqFormula} \rangle \\
\langle \textit{AndFormula} \rangle &\rightarrow \langle \textit{OrFormula} \rangle \\
&\mid \langle \textit{OrFormula} \rangle \wedge \langle \textit{AndFormula} \rangle \\
\langle \textit{OrFormula} \rangle &\rightarrow \langle \textit{SubFormula} \rangle \\
&\mid \langle \textit{SubFormula} \rangle \vee \langle \textit{OrFormula} \rangle \\
\langle \textit{SubFormula} \rangle &\rightarrow (\langle \textit{Formula} \rangle) \\
&\mid \textit{identifikátor} (\textit{identifikátor}) \\
&\mid \neg \langle \textit{SubFormula} \rangle \\
&\mid \langle \textit{Operand} \rangle \langle \textit{CompOp} \rangle \langle \textit{Operand} \rangle \\
&\mid \langle \textit{Kvantifikátor} \rangle \langle \textit{Formula} \rangle \\
\langle \textit{Kvantifikátor} \rangle &\rightarrow \exists \langle \textit{Identifikátory} \rangle \\
&\mid \forall \langle \textit{Identifikátory} \rangle \\
\langle \textit{CompOp} \rangle &\rightarrow < \mid > \mid = \mid \neq \mid \leq \mid \geq \\
\langle \textit{Operand} \rangle &\rightarrow \textit{identifikátor.identifikátor} \\
&\mid \langle \textit{MatFormula} \rangle \\
\langle \textit{MatFormula} \rangle &\rightarrow \langle \textit{MinFormula} \rangle \\
&\mid \langle \textit{MinFormula} \rangle + \langle \textit{MatFormula} \rangle \\
\langle \textit{MinFormula} \rangle &\rightarrow \langle \textit{MulFormula} \rangle \\
&\mid \langle \textit{MulFormula} \rangle - \langle \textit{MinFormula} \rangle \\
\langle \textit{MulFormula} \rangle &\rightarrow \langle \textit{DivFormula} \rangle \\
&\mid \langle \textit{DivFormula} \rangle * \langle \textit{MulFormula} \rangle
\end{aligned}
$$

$$
\begin{aligned}
< \textit{DivFormula} > \rightarrow\ & < \textit{SubMatFormula} > \\
|\ & < \textit{SubMatFormula} > / < \textit{DivFormula} > \\
< \textit{SubMatFormula} > \rightarrow\ & (< \textit{MatFormula} >) \\
|\ & \textit{číslo} \\
|\ & \textit{reťazec}
\end{aligned}
$$

Obr. 2.2 Schéma dotazu NRK

Prepisovacie pravidlo $< X > \rightarrow y \mid z$ znamená, že formula *X* môže nadobúdať tvar *y*, alebo *z*. Výrazy v lomených zátvorkách sa musia vyskytovať na ľavej strane niektorého prepisovacieho pravidla, na pravej strane sa vyskytovať môžu. *identifikátor* označuje reťazec z písmen alebo číslic, ktorý nezačína číslicou a v dotaze predstavuje názvy relácií, atribútov a premenných. *číslo* je reťazec číslic a v dotaze predstavuje celočiselné konštanty. *reťazec* je akýkoľvek reťazec znakov uzavretých v apostrofoch.

Algoritmus vyhodnocovania dotazu pomocou stromu dotazu vychádza z definície výsledku formuly NRK. Pretože podľa schémy 2.2 je koreňom stromu vždy operácia, ktorej výsledok je typu pravda/nepravda, má byť do výsledku dotazu zahrnutá tá množina n-tíc, ktorá po dosadení za voľné premenné do stromu spôsobí kladné vyhodnotenie koreňa stromu. Vyhodnocovacia funkcia ktorá sa zavolá pre koreň stromu je rekurzívna, typu post-order, čiže vyhodnocovanie uzlov v strome postupuje z najhlbších vrstiev smerom hore.

Algoritmus hľadania n-tíc, pre ktoré sa strom vyhodnotí pravdivo, je iteratívny - skúša postupne dosadiť všetky možné kombinácie n-tíc do premenných v dotaze, a pre každú takúto skupinu zisťuje, či výsledok vyhodnotenia stromu dotazu je pravda. Ak to platí, potom sú n-tice (dosadené za voľné premenné) z tejto skupiny zahrnuté do výsledku. Pred výstupom jednotlivých n-tíc sa môže na tieto n-tice aplikovať projekcia na určité atribúty. To môže viesť k tomu že výstupná relácia bude obsahovať rovnaké n-tice, čo je neprípustné a preto sa musia z výsledku odstrániť. Pokus o pridanie duplicitného riadku do výstupu aplikácia oznamuje varovaním. Dôležitou informáciou pre tento typ vyhodnotenia je množina n-tíc pre každú premennú v dotaze, inými slovami, ak má algoritmus do každej premennej dosadiť všetky možné n-tice, musí byť známa množina relácií, z ktorých sa n-tice za premennú dosadzujú. Množina relácií sa pre danú premennú určí z atomických podformulí typu predikátu (*Rel(v)*) už pri stavaní stromu dotazu. Ak sa pre rovnakú premennú vyskytuje viacero takýchto predikátových formulí, výsledná množina z ktorej sa budú n-tice za premennú dosadzovať je zjednotením relácií určených predikátovými formulami. Jediné kritérium ktoré musí byť splnené pre množinu relácií danej premennej je, že všetky relácie z množiny majú rovnakú schému relácie. V praxi to znamená že ak sa v dotaze vyskytnú napr. podformuly *HEREC_SK(h)* a *HEREC_CZ(h)*, musia tabuľky *HEREC_CZ* aj *HEREC_SK* obsahovať rovnaký počet stĺpcov rovnakého mena a typu. Poradie stĺpcov musí byť tiež rovnaké. Pri dodržaní týchto obmedzení sa za premennú *h* dosadia postupne všetky riadky z obidvoch tabuliek.

Súčasťou algoritmu vyhodnotenia je aj kontrola používania jendotlivých premenných v rôznych častiach dotazu. Tá zaisťuje, že viazané premenné použité v kvant.

podformulách nieje možné pridať k výstupu a takisto kontroluje, či pre všteky viazané premenné platí, že sa vyskytujú len vnútri formuly, v ktorej sú viazané.

Po postavení stromu a určení množín relácií pre každú premennú sa začína dosadzovať, a pre každé dosadenie do premenných sa spustí vyhodnotenie stromu. Takýchto vyhodnotení stromu môže v rámci vyhodnotenia jedného dotazu prebehnúť aj niekoľko desiatok tisíc a preto bolo nutné optimalizovať tento proces. Datové uzly, teda uzly ktoré v strome už nemajú následníkov, sa vyhodnocujú len pri prvom prechode algoritmu. To prinieslo najväčšie zrýchlenie algoritmu, keďže práve pri vyhodnotení týchto uzlov sa kontroluje správny tvar terminálov *identifikátor*, *číslo* a *reťazec* zo schémy na obr. 2.2. Pri vyhodnotení formúl typu $\exists\, v\ (\ f(v))$ hľadáme aspoň jednu n-ticu, pre ktorú po dosadení za premennú *v* formula *f(v)* platí. Keďže premenná je viazaná a nemožno ju zahrnúť do výstupu, nieje nutné poznať všetky n-tice pre ktoré formula platí, a po nájdení prvej n-tice môžeme proces dosadzovania tejto konkrétnej premennej ukončiť. Podobné pravidlo platí aj u formúl $\forall\, v\ (\ f(v))$ s tým rozdielom, že proces ukončíme po nájdení prvej n-tice pre ktorú formula neplatí. Tento typ optimalizácie je ale závislý na údajoch v tabuľkách databázy. Aj napriek optimalizáciám použitým v algoritme sa pre niektoré vstupné dotazy, využivajúce rozsiahle tabuľky z databázy, doba vyhodnotenia výrazu značne predĺži a to viedlo k riešeniu vypustenia samostatného vlákna v ktorom sa vykonáva vyhodnotenie dotazu. Tento spôsob riešenia dostatočne izoluje kód riadiaci chod užívateľského rozhrania od vyhodnocovania dotazov, a zároveň sú z pracujúceho vlákna vysielané informácie o priebehu vyhodnocovania.

3 Požiadavky na program

Aplikácia TRC je výuková aplikácia, ktorej hlavné ciele sú vyhodnotenie dotazu NRK a jeho vizualizácia. Preto je navrhnutá tak, aby okrem správneho vyhodnotenia poskytovala prostredie v ktorom je užívateľom umožnené sledovať priebeh tohto vyhodnotenia a zobrazovať štruktúru vstupných dotazov v grafickej podobe. Značenie a syntax NRK použitá pre zápis dotazov zodpovedá značeniu ktoré sa používa pri výuke predmetu Databázové systémy na MFF UK. Zo zadania sa predpokladá, že aplikácia bude používaná len na výukové účely, a preto nieje dôvod na optimalizáciu na veľké vstupy dát alebo rýchosť spracovania vstupu.

3.1 Zápis a značenie dotazu

Syntax vstupného dotazu popisuje schéma na obr. 2.2, a tvar jednotlivých formúl je definovaný v kapitole 2.2. Mierne rozdiely medzi zápisom dotazu v publikáciách [2] a [3], a zápisom v aplikácii TRC sú v značení niektorých operácií v dotaze. Dotaz je uzavretý v zložených zátvorkách ({}) a znakom |, je rozdelený na dve časti. Ľavá časť dotazu obsahuje identifikátory voľných premenných, za ktorými môže nasledovať projekcia na skupinu stĺpcov tejto premennej. Odlišnosťou je to, že projekcia v ľavej časti dotazu sa zapisuje vymenovaním stĺpcov tabuľky v hranatých zátvorkách, zatiaľ čo v pravej časti dotazu sa projekcia označuje bodkou. Tvar ľavej strany dotazu je zobrazený nasledujúcou schémou:

identifikátor, identifikátor[identifikátor, …], …

Pravá strana dotazu obsahuje podformuly oddelené operátormi. Matematické operátory sú označené klasickými znakmi (+,-,*,/), pre logické operátory existujú dva varianty. Buď sa použijú špeciálne znaky, alebo sa operácie zapíšu kľúčovým slovom. Zoznam znakov a kľúčových slov pre logické operácie:

implikácia	$\Rightarrow$	*LOGIMP*
ekvivalencia	$\Leftrightarrow$	*LOGEQ*
konjunkcia	$\wedge$	*AND*
disjunkcia	$\vee$	*OR*
negácia	$\neg$	*NOT*
existenčný kvant.	$\exists$	*EXISTS*
všeobecný kvant.	$\forall$	*FOREACH*

Pred vyhodnotením dotazu sa špeciálne znaky logických operácií nahradia im zodpovedajúcimi kľúčovými slovami a až potom sa začne samotné vyhodnotenie. Projekcia sa vo formulách v pravej časti dotazu značí bodkou, pred ktorú sa uvedie identifikátor premennej, a za ktorú sa napíše jeden identifikátor atribútu.

Na porovnávanie slúžia operátory:

=, <, >,

≤ značíme <=,
≥ značíme >=,
≠ značíme dvojicou znakov <>

Na označenie premenných sa vzhľadom na konvenciu odporúčajú použiť malé písmená, aby nedochádzalo k zámene s identifikátormi tabuliek, ktoré sa značia veľkými písmenami. Ďalšie odporúčanie sa týka zátvorkovania jednotlivých podformúl, ktoré je vhodné používať hlavne pri zložitejších dotazoch. Ak sa v jednej formule vyskytuje viacero operácií, ich priorita sa určuje podľa nasledujúcej tabuľky, kde operácie vo vyšších riadkoch majú vyššiu prioritu.

⇒, ⇔
∨, ∧
∀, ∃, ¬
porovnávacie operácie
projekcia

Pre formuly s kvantifikátormi sa odporúča uzátvorkovať celú podformulu, alebo časť za deklaráciou kvant. premenných takto:

∃ v (f(v)), alebo *(∃ v f(v))*

Najmenšou jednotkou v rámci dotazu je *identifikátor*, *číslo* a *reťazec*. Identifikátorom značíme premenné, relácie a atribúty a môže obsahovať písmená, číslice a znak '_', nesmie začínať číslicou. Celé číslo vyjadríme skupinou číslic, ale čísla s desatinou časťou sa musia zapísať pomocou reťazca, teda musia byť uzavreté v apostrofoch. Takto zapísané desatiné čísla môžu obsahovať ľubovoľný počet miest pred, a za desatinou čiarkou. Reťazcom sa okrem desatiných čísel a textu vyjadrujú ešte hodnoty typu dátum, ktoré musia byť zapísané v tvare dd.MM.yyyy, teda dvojčíslím vyjadrený deň, mesiac a štvročíslím vyjadrený rok, pričom jednotlivé údaje musia byť oddelené bodkou.

Medzi jednotlivými znakmi operácií, zátvorkami a identifikátormi sa môže vyskytovať ľubovoľný počet bielych, znakov ktoré sa pred vyhodnotením odstránia. Vďaka tomu je možné formátovať vstupný dotaz a umiestňovať niektoré zložitejšie formuly na riadok samostatne.

3.2 Vizualizácia dotazu

Požiadavke na vizualizáciu najlepšie zodpovedá znázornenie stromu dotazu. Binárny strom dotazu, ktorý je použitý na vizualizáciu, je efektívne využitý aj pri vyhodnocovaní dotazu a pozostáva z uzlov dvojakého typu, ktoré sú farebne odlíšené. Sú to uzly operácií, a datové uzly. Každý uzol v strome má svoj vnútorný datový typ, ten

predstavuje typ výsledku pri operačných uzloch, pre datové uzly je to typ informácie ktorú uchovávajú. Uzly majú jeden z nasledujúcich typov:

- *bool* – tohto typu sú operačné uzly logických operácií, ktoré majú hodnoty pravda, alebo nepravda.
- *number* – typ nadobúdajú datové uzly alebo uzly matematických operácií. Predstavuje celé číslo, alebo desatiné číslo zadané v reťazci.
- *date* – typ dátum nadobúdajú iba datové uzly.
- *string* – typ nadobúdajú datové uzly, alebo operačný uzol operácie +, ktorou je možné spájať jednotlivé reťazce.
- *variable* – označuje typ, ktorý majú datové uzly reprezentujúce identifikátor premennej, relácie, alebo atribútu relácie.
- *variables* – tento typ majú datové uzly reprezentujúce identifikátory premenných za kvantifikátorom pri skrátenom zápise kvantifikovaných výrokov.

Datové uzly, ktoré sú v aplikácii označené štandardne sivou farbou, reprezentujú identifikátory premenných, atribútov a relácií, a konštanty. Datové uzly nemajú v strome následníkov a tvoria poslednú hladinu stromu. Popisky na týchto uzloch zodpovedajú ich obsahu a typu, teda vyjadrujú číselnú, alebo textovú konštantu, dátum alebo identifikátor. Operačné uzly majú v strome následníkov, ktorí reprezentujú operandy operačného uzlu. Tieto operandy môžu byť podľa druhu operácie buď uzly datové, alebo operačné a ich hodnota sa určí až výpočtom. Po označení ľubovoľného z uzlov je užívateľovi poskytnutý typ uzlu, dátový typ jeho hodnoty, aktuálna hodnota, a časť dotazu ktorú uzol reprezentuje. Pre operačné uzly sa vypíše aj typ operácie.

Ďalšou požiadavkou na aplikáciu je, aby užívateľ mohol proces vyhodnotenia dotazu sledovať. Musí byť možné pre každé dosadenie súboru n-tíc do premenných sledovať výsledky ľubovoľnej operácie v strome. Túto funkciu zabezpečuje v aplikácii TRC vyhodnocovanie v špeciálnom móde, ktorý umožňuje krokovanie vyhodnocovania. Pre každú premennú je možné sledovať, ktorá n-tica je do tejto premennej práve dosadená, a hodnoty atribútov tejto n-tice. Krokovanie funguje na princípe breakpointov(zarážok), to znamená, že ak si užívateľ praje pozastaviť proces vyhodnocovania v určitom uzle, nastaví pre tento uzol breakpoint. Po príchode vyhodnocovacieho procesu na takto označený uzol je možné okrem hodnôt premenných sledovať aj hodnoty a výsledky všetkých uzlov v strome. Breakpointy je možné nastavovať len pre operačné uzly, ktorých hodnota sa počíta z ich následníkov. Počet uzlov pre ktoré je možné nastaviť breakpoint nieje obmedzený, a breakpointy sa môžu pridávať aj odoberať počas behu vyhodnocovacieho procesu. Ak v strome nebude nastavený žiaden breakpoint, proces nebude zastavovať, a budú sa len zobrazovať aktuálne dosadenené n-tice pre premenné. Ďalšou vlastnosťou vyhodnocovania v krokovacom móde je priebežné dopĺňanie výstupnej relácie. Ak počas krokovania dôjde k vyhodnoteniu koreňa stromu a jeho pravdivostná hodnota bude pravda, všetky n-tice aktuálne dosadené do voľných premenných sa pridajú do výstupnej relácie. V prípade, že vo výstupnej relácii už taká n-tica je, potom sa taká n-tica do výsledku nepridá a tento pokus je oznámený užívateľovi.

4 Užívateľská dokumentácia

4.1 Inštalácia

Aplikácia TRC je určená pre operačný systém Windows, a pre verzie 98 a vyššie. Pred spustením aplikácie je nutné mať nainštalovaný balík Microsoft .NET Framework 2.0. Aplikácia sa inštaluje pomocou spustiteľného inštalačného súboru. Sú dve varianty inštalačného súboru. Je to inštalácia obsahujúca .NET Framework 2.0, alebo len inštalácia samotnej aplikácie. Inštalácia s .NET-om sama detekuje, či je na počítači nainštalovaná verzia .NET 2.0, a ak je to potrebné, túto verziu nainštaluje. Počas inštalácie môže užívateľ vybrať miesto, kde sa aplikácia nainštaluje, a zvoliť vytvorenie položky v štart menu odkiaľ je možné program po inštalácii spustiť. Spolu s programom *TRC.exe* sa nainštalujú aj ukážkové projekty a tabuľky, ktoré sú umiestnené v priečinku *examples* v adresári aplikácie. Dôležitou knižnicou ktorú aplikácia používa je *MagicLibrary.dll*, tá sa nachádza v adresári aplikácie a je nutná pre správny chod programu.

4.2 Typy súborov

Aplikácia TRC pracuje s dvomi typmi súborov:

- súbor projektu má príponu *trc* a obsahuje zoznam dotazov s poznámkami, a databázu vo forme tabuliek. Tento súbor je binárny, a užívateľ môže v aplikácii TRC takéto súbory vytvárať, editovať a ukladať zmeny.
- súbor tabuľky s príponou *tab* predstavuje tabuľku v databáze. Uložené sú informácie o typoch a názvoch jednotlivých stĺpcov, ako aj samotné údaje v riadkoch tabuľky. Súbor tabuľky je typu XML.

Oba typy súborov je možné otvoriť pomocou menu v aplikácii TRC, alebo funkciou drag&drop systému windows, ktorým je možné otvoriť naraz viacero súborov tabuliek, alebo jeden súbor projektu. Celé okno aplikácie slúži ako cieľová plocha pre drag&drop projektu. Plocha pre drag&drop súborov tabuliek je tvorená plávajúcim oknom *Tables*, ktoré slúži na prácu s tabuľkami. Po otvorení súboru projektu môže užívateľ aj naďalej otvárať súbory s tabuľkami a doplňať tak aktuálnu databázu aplikácie.

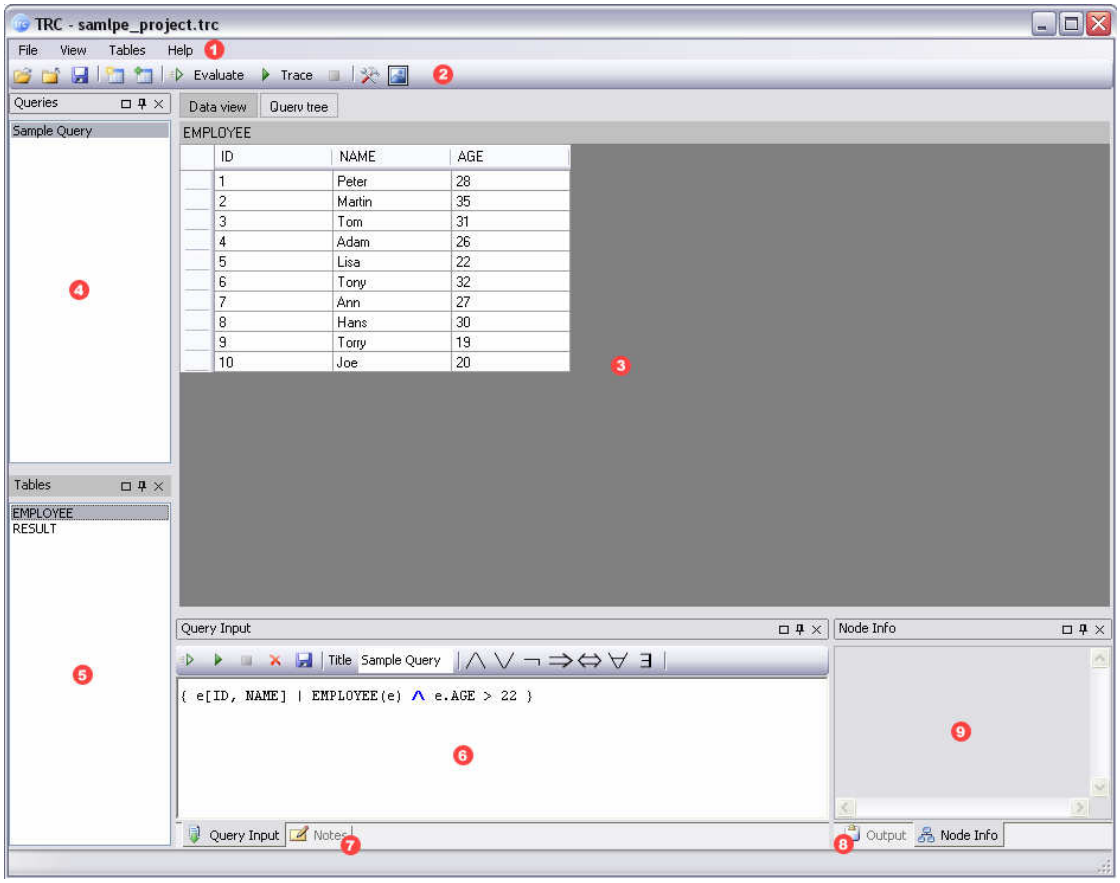
Osobitným spôsobom otvárania súborov je použitie príkazového riadku, z ktorého je možné otvoriť vždy len jeden súbor príkazom v tvare *trc.exe -p projectfile.trc* pre súbor projektu, alebo *trc.exe -t tablefile.tab* pre súbor tabuľky. Po správnej inštalácii sa pomocou registrov systému windows asociujú súbory *.trc a*.tab s aplikáciou TRC a preto je možné otvoriť súbor aj dvojkliknutím na ikonu súboru, ktorá má po inštalácii rovnakú ikonu ako aplikácia TRC.

Neodporúča sa meniť vnútornú štuktúru týchto súborov a editovať ich pomocou iných nástrojov ako je aplikácia TRC. Prípadnú zmenu v štruktúre, alebo poškodenie súboru aplikácia zistí a ohlási chybu pri otváraní tohto súboru.

4.3 Podrobný popis pracovného prostredia

Užívateľské prostredie aplikácie TRC je typu SDI (single document interface), to znamená, že umožňuje pracovať naraz len s jedným projektom a vyhodnocovať, alebo krokovať len jeden dotaz. Pre súčasnú prácu s viacerými projektami alebo dotazmi môže užívateľ spustiť novú inštanciu aplikácie.

Okno aplikácie je členené na hlavné menu, toolbar, stavový riadok a zobrazovací panel, ktorý tvorí zvyšnú časť okna. Na prácu s dotazmi, tabuľkami, a na zobrazovanie iformácií o procese vyhodnocovania slúžia plávajúce okná, ktorým užívateľ môže definovať veľkosť podľa potreby a presunúť ich na ľubovoľné miesto v hlavnom okne, alebo skryť nepotrebné plávajúce okno.



Obr. 4.3 Okno aplikácie TRC

Hlavné menu aplikácie*(č.1 na Obr. 4.3)* obsahuje základné príkazy na prácu s projektom a tabuľkami. Špeciálne príkazy ako premenovanie a editácia dotazov alebo tabuliek sú už ale súčasťou plávajúcich okien. Menu je rozdelené na štyri časti a obsahuje tieto položky:

File menu

- *Save project* – uloží práve otvorený projekt do súboru. Táto položka je prístupná, len ak editujeme súbor projektu.
- *Save project as* – uloží novo vytvorený, alebo editovaný projekt do súboru. Pri ukladaní je možné zvoliť umiestnenie aj názov cieľového súboru. Položka v menu je prístupná len ak je čo uložiť, teda ak je v projekte uložená aspoň jedna tabuľka alebo dotaz.
- *Load project* – otvorí súbor projektu s príponou trc. Ak je súbor poškodený, program oznámi chybu a súbor nenačíta. Klávesová skratka pre túto položku je Ctrl + O.
- *Close project* – zavrie otvorený súbor bez uloženia. Položka je prístupná len ak je otvorený nejaký súbor projektu.
- *Options* – otvorí špeciálne okno kde je možné nastaviť vzhľad prostredia a vlastnosti vyhodnocovania dotazov. Kláv. skratka je Ctrl + P.
- *Exit* – zavrie hlavné okno a ukončí aplikáciu TRC.

View menu – pomocou tohto menu užívateľ definuje viditeľnosť plávajúcich okien:

- *Table list* – okno slúži ako zoznam všetkých tabuliek v aktuálnej databáze.
- *Output list* – do tohto okna sa vypisujú informácie o priebehu vyhodnocovania a prípadné chyby, ktoré počas vyhodnocovania nastali.
- *Node info* – po označení uzlu v strome sa do tohto okna vypíšu jeho vlastnosti a hodnota
- *Query input* – základné okno pre zadávanie vstupných dotazov. Toto okno obsahuje aj toolbar z ktorého je možné editovať a uložiť dotaz, a tiež spustiť jeho vyhodnotenie.
- *Query list* – je plávajúce okno so zoznamom všetkých uložených dotazov s poznámkami.
- *Notes* – okno obsahuje poznámku vždy pre aktuálne vybratý dotaz.
- *Show All* – zobrazí všetky plávajúce okná. Kláv. skratka je Ctrl + R.
- *Hide All* – skryje všetky zobrazené plávakúce okná a tým maximalizuje hlavný panel aplikácie. Kláv. skratka je Ctrl + H.

Tables menu

- *Add new table* – otvorí formulár pre pridanie novej tabuľky.
- *Load from file* – umožní vybratie súboru tabuľky, ktorá sa načíta do aktuálnej databázy.

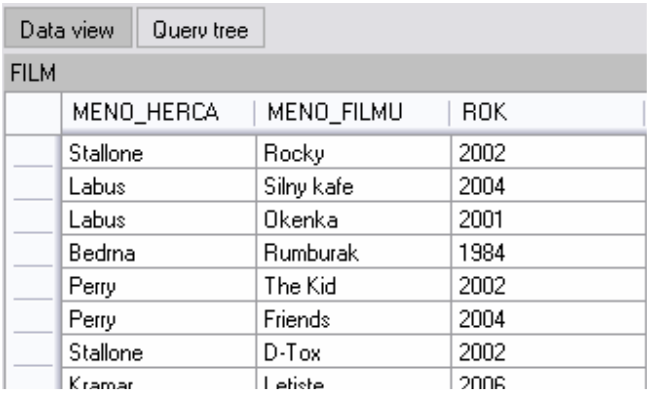
Help menu

- *TRC help* – táto položka zobrazí užívateľskú príručku k programu TRC. Príručka je vo forme kompilovanej pomoci pre systém windows (*.chm), a je uložená v adresári aplikácie – súbor TRC_help.chm. Ak tento súbor neexistuje, alebo nieje umiestnený v adresári z ktorého sa aplikácia spúšťa, pri pokuse o prístup k tomuto súboru aplikácia ohlási chybu.
- *About* – otvorí okno s informáciou o autorovi a o verzii programu.

Hlavný toolbar *(č.2 na Obr. 4.3)* obsahuje najčastejšie používané príkazy z hlavného menu: *Load project*, *Close project*, *Save project as*, *Add new table*, *Edit table* a *Options*. Na vyhodnotenie dotazu slúžia položky *Evaluate query*, *Trace query* a *Stop evaluation* z toolbaru umiestneného v okne *Query input* pre zadávanie dotazu. Toolbar obsahuje aj položku *Export to image* na uloženie stromu dotazu do obrázku spolu s textom dotazu.

Hlavný panel *(č.3 na Obr. 4.3)*, ktorý zaberá najväčšiu časť okna slúži na zobrazovanie tabuliek v databáze *(režim Data View)* a na zobrazenie stromu dotazu *(režim Query Tree)*. Prepínanie medzi týmito režimami zobrazenia umožňujú tlačidlá v ľavom hornom rohu panelu, a na prepnutie je možné použiť aj klávesovú skratku Ctrl + Tab.

Panel v režime *Data View* zobrazuje tabuľky automaticky, po vyznačení niektorej z tabuliek v plávajúcom okne *Tables*. Okrem samotných dát sa zobrazí v hornom titulkovom pruhu aj názov tabuľky.

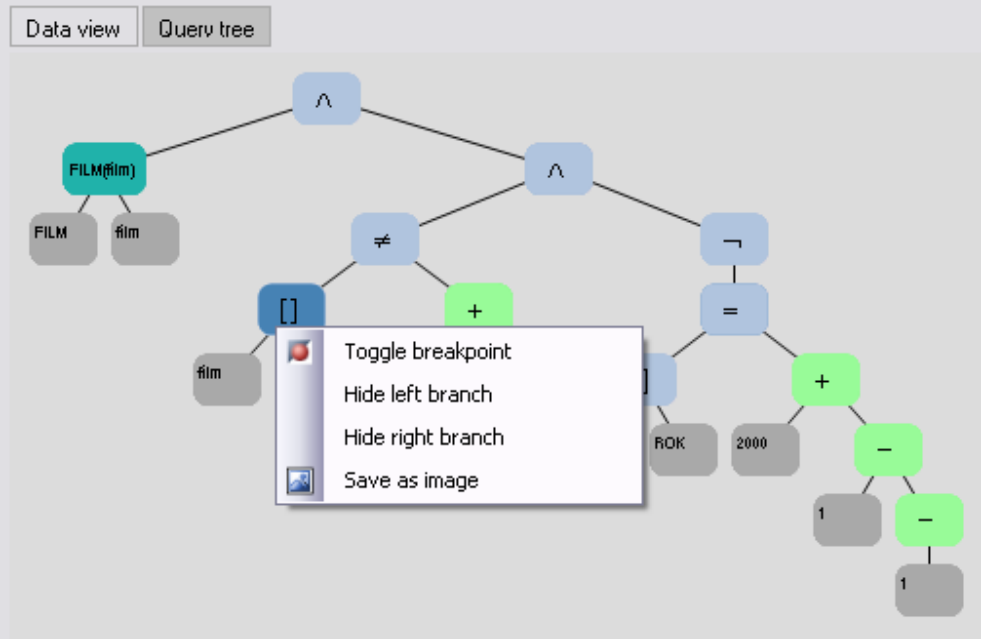


Obr. 4.3.1 Hlavný panel – režim DataView

Panel v režime *Query Tree* zobrazuje po úspešnom vyhodnotení dotazu jeho strom. Značenie stromu dotazu je popísané v kapitole 3.1. Jednotlivé typy uzlov sú farebne odlíšené a v nastaveniach programu je možné špecifikovať farbu datových uzlov, operačných uzlov logických a matematických operácií. Osobitnou farbou sú vyznačené predikátové operačné uzly – uzly predstavujúce atomickú formulu tvaru *Rel(v)*. Okrem farby uzlov je možné nastaviť aj farbu pozadia stromu a štýly písma, ktorými sú informácie o obsahu uzlov a popisky na uzloch zobrazované. Ak je strom dotazu rozsiahly a niektorý z rozmerov stromu presahuje rozmery panelu, zobrazia sa na okrajoch panelu posuvné lišty, ktorými sa dá pozícia stromu v okne nastaviť.

Rozsiahlym stromom je možné pohybovať aj použitím myši. Po stlačení prostredného tlačidla myši sa štandardný kurzor zmení a stromom je možné posúvať pohybom myši.

Pomocou myši sa v strome dajú označiť jednotlivé uzly. Po kliknutí na ľubovoľný uzol sa tento uzol farebne vyznačí, a informácie o tomto uzle sú k dispozícii v okne Node Info. Farbu vybratých uzlov môžeme zvoliť v nastavení pre každý typ uzlu. Skupina uzlov sa v strome vyznačí postupným označením jednotlivých uzlov pri stlačenej klávese Shift.



Obr. 4.3.2 Hlavný panel – režim Query Tree

Panel v režime zobrazenia stromu obsahuje kontextové menu ktoré sa objaví po kliknutí pravého tlačidla myši. Obsah tohto menu sa líši podľa toho, či menu vyvoláme pre uzol v strome, alebo pozícia myši pri kliknutí je mimo akýkoľvek uzol. Menu môže obsahovať tieto položky:

- *Toggle breakpoint* – Položka v menu je prístupná len po vyvolaní menu pre operačný uzol stromu dotazu. Po zvolení tejto položky sa na označenom uzle nastaví breakpoint. Ak tento uzol už má nastavený breakpoint potom tento príkaz breakpoint zruší. Uzol na ktorom je nastavený breakpoint je označený malou šípkou v pravej dolnej časti uzlu. Delete all breakpoints – položka v menu je prístupná len po vyvolaní menu pre ľubovoľný uzol, a len v prípade, že niektoré uzly v strome majú nastavené breakpointy. V takom prípade sa týmto príkazom všetky breakpointy v strome zrušia.

- *Hide left/right branch* – skrytie ľavého/pravého podstromu. Umožňuje skryť nezaujímavé alebo príliš rozsiahle časti stromu, ktoré sa nezmestia na plochu

Tree View panelu. Skryť, alebo zobraziť ľavý a zároveň pravý podstromu operačného uzlu v strome je možné aj dvojkliknutím na daný uzol. Pokračovanie stromu na uzle so skrytými vetvami je naznačené troma bodkami v jeho spodnej časti. Pri krokovaní vyhodnotenia sa táto položka menu objaví len v prípade, že skrytím vetvy sa neskryje aj uzol, na ktorom je nastavený nejaký breakpoint.

- *Save as image* – táto položka kontextového menu je prístupná len po vyhodnotení dotazu a vykreslení stromu dotazu. Umožňuje uložiť obrázok stromu do súboru. Po vybratí tejto položky z menu sa otvorí dialóg , kde má užívateľ možnosť vybrať umiestnenie, názov a formát súboru. Dostupné sú formáty *bmp, gif, jpg, png, tiff*. Veľkosť obrázku je rovnaká ako skutočná veľkosť stromu, a farby použité v obrázku sú rovnaké ako pri zobrazení stromu v paneli, a je možné ich meniť. V nastaveniach aplikácia si užívateľ môže zvoliť či na obrázku stromu bude zobrazený aj text dotazu, polohu, font a farbu tohto textu. Klávesová skratka pre túto funkciu je Ctrl + I.

Stavový riadok v dolnej časti okna aplikácie zobrazuje pri vyhodnocovaní dotazu stav vyhodnotenia percentuálne, pomocou ukazateľa postupu (progressbar). Vďaka tejto informácii môže užívateľ odhadnúť čas potrebný na vyhodnotenie dotazu.

Plávajúce okná

Na prácu s dotazmi a tabuľkami využíva aplikácia TRC plávajúce okná. Použitie tohto typu okien dáva užívateľovi možnosť prospôsobiť si vzhľad celého okna aplikácie, a zvoliť vlastné rozmiestnenie týchto okien. Jednotlivé plávajúce okná je možné zoskupovať do kariet potiahnutím okna za jeho titulkový pruh, a presunutím nad iné okno. V jednej karte je viditeľné vždy len jedno z okien ktoré karta obsahuje, a prepínať medzi nimi možno pomocou kliknutia na titulok okna v dolnej časti karty. Samostatné okná alebo karty je ďalej možné prichytávať k okrajom hlavného okna aplikácie.

Ak si užívateľ nepraje mať niektoré okno zobrazené, môže ho skryť pomocou položky View z hlavného menu, alebo použitím kontextového menu plávajúcich okien, ktoré sa zobrazí po kliknutí prvým tlačidlom myši na titulkový pruh niektorého z plávajúcich okien. Toto menu obsahuje zoznam všetkých okien s vyznačením ich viditeľnosti, a položky *Show all* a *Hide all* ktoré sa používajú na zobrazenie, alebo skrytie všetkých okien naraz. Rozloženie ako aj veľkosť a viditeľnosť všetkých plávajúcich okien si aplikácia pamätá. Znamená to, že ak si užívateľ nadefinuje novú polohu okien alebo zmení ich veľkosť, po znovuspustení aplikácie sa zachová usporiadanie okien. Táto funkcia programu používa špeciálny konfiguračný súbor *dock_config.dcg*, ktorý sa po úspešnej inštalácii nachádza v pomocnom adresári pre aplikácie a v jeho podadresári TRC. Najčastejšie to je cesta *Documents and Settings\User\Application Data\TRC*. Pre správny chod aplikácie nieje prítomosť tohto súboru nutná. Ak pri štarte aplikácia nenájde súbor *dock_config.dcg*, nastaví sa počiatočná pozícia okien.

Plávajúce okná aplikácie TRC:

- Table list
- Query list
- Query input
- Notes
- Node info
- Output list
- Variable watch
- Watch

Table list *(č.5 na Obr. 4.3)* je okno na prácu s tabuľkami. Každá spustená inštancia aplikácie TRC má vlastnú databázu pozostávajúcu z tabuliek. Názvy všetkých tabuliek tejto databázy sú zobrazené v zozname Table list. Užívateľ môže pridať tabuľku do tohto zoznamu vytvorením novej tabuľky, alebo načítaním tabuľky zo súboru. Otvorením súboru s projektom sa tento list tabuliek naplní tabuľkami, ktoré sú uložené v súbore projektu. Toto okno podporuje drag&drop pre tabuľkové súbory s príponou *tab*. Po označení názvu ktorejkoľvek tabuľky sa zobrazovací panel aplikácie prepne do režimu *DataView* a zobrazí túto tabuľku.

Kontextové menu okna Table list obsahuje tieto položky:

- *New* – otvorí formulár pre vytvorenie novej tabuľky.
- *Edit selected* – otvorí formulár pre editovanie práve vybratej tabuľky. Položka je dostupná, len ak je v zozname vybratá nejaká tabuľka.
- *Delete selected* – vymaže označenú tabuľku zo zoznamu. Označenú tabuľku môže užívateľ zmazať aj pomosou klávesy Delete.
- *Load from file* – otvorí dialóg pre načítanie tabuľky zo súboru.
- *Save to file* – otvorí dialóg pre uloženie vybratej tabuľky do súboru.

Špeciálnou tabuľkou v tomto zozname je tabuľka *RESULT*, ktorá je výsledkom aktuálne vyhodnoteného dotazu. Túto tabuľku nieje možné editovať, a tiež nieje možné vytvoriť inú tabuľku s takýmto menom.

Query input *(č.6 na Obr. 4.3)* je okno určené na zadávanie a zobrazovanie dotazu. Obsahuje toolbar v hornej časti, a zvyšnú časť zaberá textové okno do ktorého sa píše samotný dotaz. Štýl, veľkosť a farba písma pre toto textové okno sa dá zmeniť vo formulári nastavení. Syntax a značenie dotazu je popísané v kapitole 3.1.

Toolbar okna Query input obsahuje tieto položky:

- *Evaluate query* – spustí vyhodnotenie vloženého dotazu.
- *Trace query* – spustí krokovanie vyhodnoteného dotazu. Položka je prístupná len v prípade, že vyhodnotenie dotazu prebehlo úspešne.
- *Stop evaluation* – zastaví proces vyhodnocovania, alebo krokovania dotazu. Položka je prístupná len ak je proces vyhodnocovania alebo krokovania aktívny.

- *Clear query input* – pomocou tohto tlačidla užívateľ vymaže aktuálne editovaný dotaz z okna *Query input*, vymaže prípadne jeho výsledok (tabuľka *RESULT*) a aj strom ak bol dotaz vyhodnotený, a pripraví okno pre zadanie nového vstupu. Ak je program v režime krokovania dotazu, po zvolení tohto príkazu sa krokovanie okamžite ukončí. Neuložený dotaz sa po stlačení tlačidla clear query input nenávratne vymaže.
- *Title* – textbox pre vloženie mena dotazu, toto meno sa zobrazí po uložení dotazu v okne *Query list*. Ak pri uložení dotazu je tento titulok prázdny, ako titulok dotazu v okne *Query list* sa použije text dotazu. Odporúča sa použiť vždy titulok, ktorý popisuje výsledok dotazu.
- *Save query* – tlačidlo na uloženie dotazu. Po uložení dotazu sa titulok tohto dotazu pridá do zoznamu okna *Query list* a dotaz je dočasne uložený. Pre uloženie do súboru je nutné ešte uložiť dotaz v rámci projektu. Ak chceme uložiť zmeny v dotaze, ponecháme titulok dotazu v textboxe *Title* rovnaký a stlačíme tlačidlo Save. Po vybratí položky save sa spolu s textom dotazu uloží aj poznámka k tomuto dotazu. Položka Save sa zobrazí aj v kontextovom menu okna *Query input*.
- *Tlačidlá pre vloženie symbolov logických operácií* – pre lepšiu orientáciu v texte dotazu môže užívateľ použiť namiesto kľúčových slov operácií symboly. Symboly operácií možno vložiť aj pomocou kontextového menu okna *Query input*.

Queries list *(č.4 na Obr. 4.3)* obsahuje zoznam uložených dotazov. Po označení dotazu sa v okne *Query input* zobrazí text dotazu, a poznámka uložená spolu s dotazom je zobrazená v okne *Notes*.

Kontextové menu okna obsahuje položky:

- *Delete* – vymaže označený dotaz zo zoznamu. Pre vymazanie dotazu sa dá použiť aj klávesa Delete.
- *Rename* – premenuje titulok dotazu. Na premenovanie titulku sa používa len tento príkaz. Zmena titulku v okne *Query input* a následné uloženie dotazu, spôsobí, že sa do zoznamu dotazov pridá nový dotaz s rovnakým textom, ale zmeneným titulkom.

Okno **Notes** *(č.7 na Obr. 4.3)* slúži na pridanie a úpravu textovej poznámky k dotazu. Po uložení pomocou príkazu z kontextového menu tohto okna sa pridá poznámka k dotazu.

Node info *(č.9 na Obr. 4.3)* zobrazuje informácie o uzloch v strome. Po označení nejakého uzlu v strome dotazu sa premiestni toto okno do popredia a zobrazí informáciu o vlastnostiach tohto uzlu. O uzle sa zobrazí jeho typ, datový typ, hodnota a časť dotazu ktorú reprezentuje. Informácia je tvaru:

- *Node type* – typ uzlu. Hodnota môže byť *DATA NODE*, alebo *OPERATION*

- *Data type* – datový typ hodnoty uzlu. Podrobný zoznam a popis jednotlivých datových typov obsahuje kapitola 3.2.
- *Value* – hodnota uložená v uzle. (Zodpovedá datovému typu.)
- *Query part* – časť dotazu ktorú uzol reprezentuje. Túto informáciu majú len operačné uzly. Časť dotazu ktorú reprezentuje datový uzol je jeho hodnota.

Okno **Output** *(č.8 na Obr. 4.3)* priebežne zobrazuje informácie o procese vyhodnocovania. Ak sa pri vyhodnocovaní vyskytne chyba, vyhodnotenie sa zastaví a chyba je zobrazená v tomto okne. Zoznam správ ktoré sa môžu zobraziť v okne Output:

- *tree build successful* – oznámenie o úspešnom postavení stromu dotazu.
- *evaluation successful* – ukončenie vyhodnocovania.
- *evaluation stopped* – oznámenie o zastavení vyhodnocovania užívateľom.
- *process suspended* – oznámenie prerušenia vyhodnocovania užívateľom.
- *process resumed* – obnovenie vyhodnocovania po pozastavení.
- *query is safe* – oznámenie o bezpečnosti vyhodnocovanej formuly.
- *query is not safe*
- *row duplicity, row was not added to output* – oznámenie o pokuse pridať duplicitný riadok do výsledku dotazu pri krokovaní.
- *Error: Typ chyby. [riadok, stĺpec]* – ohlásenie chyby. Za slovom error nasleduje text chyby a za ním v hranatých zátvorkách pozícia chyby v dotaze. Pozícia je pri chybe špecifikovaná len v prípade že ide o určitý druh syntaktickej chyby.

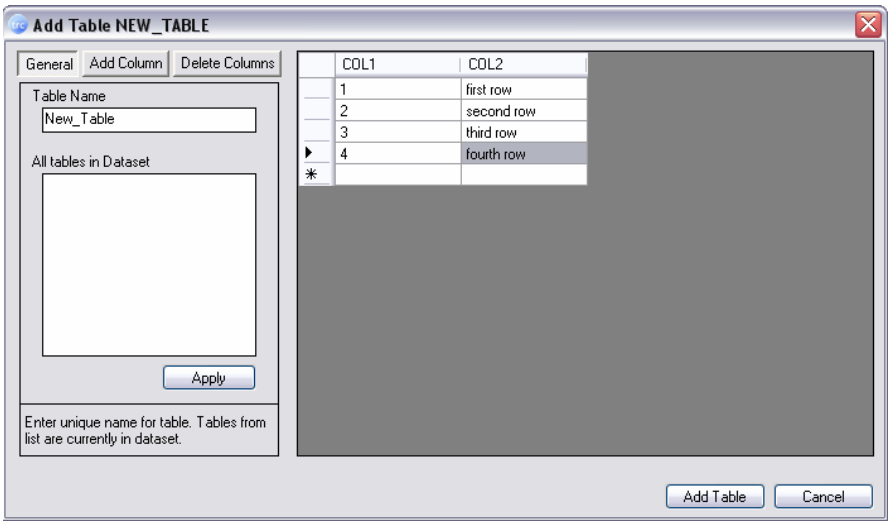
Zoznam chýb ktoré sa môžu vyskytnúť pri vyhodnocovaní výrazu a ich význam:

- *"String in query is not closed, or newline character occured in string."* – chyba nastane ak je v dotaze nesprávne uzavretý reťazec, alebo sa v reťazci vyskytuje odriadkovanie.
- *"Operation %s operand missing."* – chyba nastane ak operácii typu *%s* chýba aspoň jeden operand.
- *"Operation %s operand not proper."* – operand operácie typu *%s* má nesprávny datový typ, alebo hodnotu.
- *"Invalid DataSet. Cannot evaluate query."* – chyba nastane ak je poškodená aktuálna databáza aplikácie.
- *"Table %s is not available in connected database."* – chyba nastane ak sa v dotaze použije neexistujúca tabuľka s názvom *%s*.
- *"Token %s is not variable, string or number."* – chyba nastane ak je token *%s* použitý na mieste, kde je očakávaný výskyt premennej, čísla, alebo reťazca.
- *"Token %s is not variable."* – chyba nastane ak je token *%s* použitý na mieste, kde je očakávaný výskyt premennej.
- *"Query contains more definitions of variable %s ."* - chyba nastane ak je premenná *%s* definovaná v dotaze viackrát, napr. vyskytuje sa ako viazaná aj voľná premenná.

- *"Variable %s value is not assigned."* – pre premennú *%s* nieje definovaná ani jedna relácia a preto nemôže byť jej hodnota určená.
- *"Unsupported data type loaded from database."* – hodnota z databázy má nesprávny datový typ.
- *"Unsupported operand data type for operation %s."* – operand operácie typu *%s* má nesprávny datový typ.
- *"Different operand types can not be converted."* – operandy niektorej z operácií nemujú rovnaký datový typ.
- *"Domain of variable %s is not specified."* – pre premennú *%s* nieje definovaná ani jedna relácia.
- *"Invalid left side of query. Allowed format is 'identifier[col_id, col_id, ...]'."* – chyba nastane ak má ľavá strana dotazu nesprávny formát.
- *"Column %s is not valid for specified variable."* – chyba nastane pri pokuse o projekciu premennej na stĺpec *%s* ktorý nieje atribútom relácie pre túto premennú.
- *"Zero division error occured."* - chyba nastane ak sa v dotaze, alebo pri vyhodnotení dotazu, vyskytne delenie nulou.
- *"Operation result is not boolean type."* – chyba nastane ak operácia ktorá predstavuje koreň stromu dotazu je iného typu ako boolean.
- *"Cell with null value in DB table %s."* - chyba nastane pri pokuse dosadiť do premennej n-ticu ktorá obsahuje prázdnu bunku.
- *"Number overflow error."* – chyba oznamuje pretečenie celého, alebo desatinného čísla.
- *"Incompatible relation schema of %s relations."* – chyba nastane pri pokuse priradiť jednej premennej relácie s odlišnou schémou.
- *"Variable %s used out of its scope."* – chyba oznamuje použitie viazanej premennej mimo oblasť jej definície.
- *"Query syntax error."* – bližšie nešpecifikovaná chyba syntaxe dotazu. Napr. dotaz nieje uzavretý zloženými zátvorkami alebo je zle uzátvorkovaný.

4.4 Vytvorenie novej tabuľky

Na vytvorenie novej tabuľky slúži špeciálny formulár, ktorý môžeme vyvolať príkazom *Add New Table*, z hlavného menu, alebo kontextového menu plávajúceho okna *Tables*. Formulár je členený na časť nastavení vľavo, a časť pre zobrazenie dát v tabuľke.



Obr. 4.4 Formulár Add Table

V sekcii všeobecých nastavení (*General*), sa vyplní názov tabuľky. Pre názvy tabuliek sa používajú veľké písmená, a preto bude aj názov zadaný malými písmenami skonvertovaný na veľké. Ako pomôcka pri zadávaní mena tabuľky slúži zoznam už prítomných tabuliek v databáze, keďže názov tabuľky musí byť unikátny. Špeciálnym obmedzením je to, že reťazec *RESULT* nemožno použiť ako názov novej tabuľky, pretože to je vyhradený názov pre výsledky dotazov.

Nové stĺpce tabuľky sa definujú v časti *Add Column*. Po zadaní unikátneho názvu stĺpca, ktorý bude skonvertovaný na veľké písmená môže užívateľ zvoliť jeden z dostupných datových typov:

- *DATE* – datový typ dátum, hodnoty sa zadávajú v tvare *dd.MM.yyyy*, pre zadanie hodnoty v dotaze sa musí dátum uzatvoriť do reťazca.
- *FLOAT* – desatinné číslo
- *INT* – celé číslo
- *STRING* - reťazec

Stĺpce tabuľky je možné zmazať v časti *Delete columns*. Zo zoznamu stĺpcov sa vyberú stĺpce na zmazanie a po stlačení tlačidla Delete sa spolu s obsahom stĺpce vymažú.

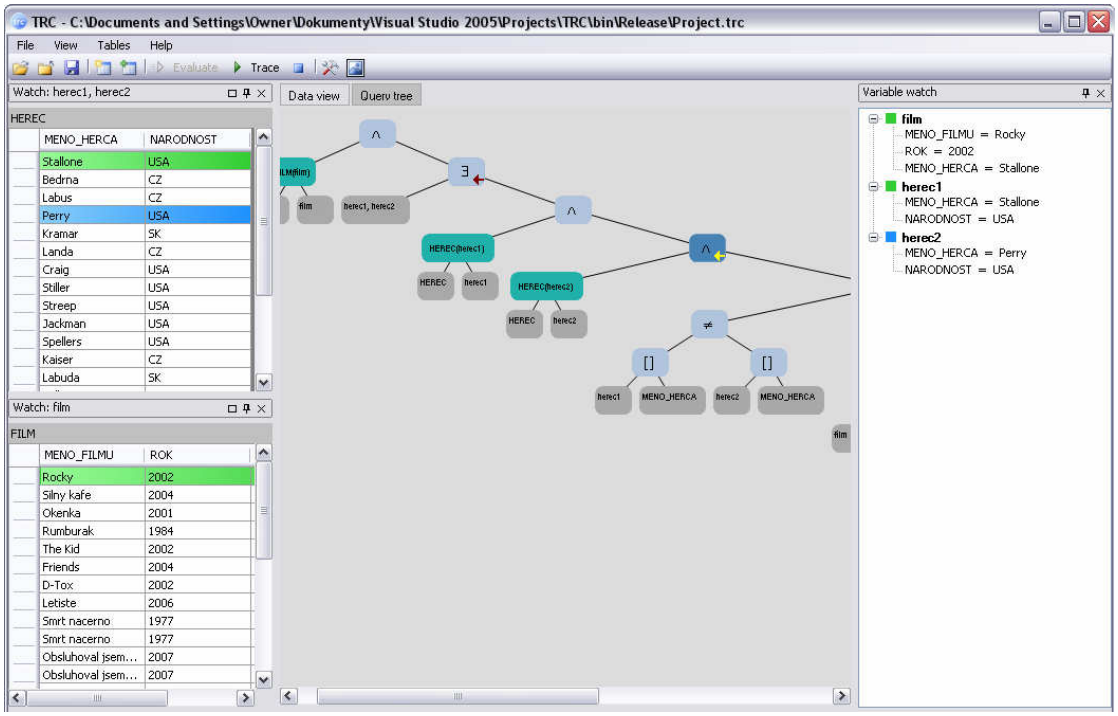
V časti pre zobrazenie tabuľky sa po nadefinovaní stĺpcov dajú vkladať údaje do jednotlivých buniek kliknutím na bunku a následným vložením hodnoty. Formát vkladaných údajov musí zodpovedať typu stĺpca. Pri zadávaní číselných hodnôt je možné zadať aj znamienko. Reťazce nemusia byť uzavreté v apostrofoch ako pri zadávaní reťazcov v dotaze. Žiadna bunka v tabuľke nesmie byť prázdna. Pri zadaní nesprávneho formátu dát (napr. zadaním reťazca do stĺpca s datovým typom *INT*), prázdnej bunky, alebo pri zadaní rovnakého riadku ohlási aplikácia varovanie a nesprávny údaj nieje akceptovaný. Datový typ jednotlivých stĺpcov v tabuľke je uvedený za názvom stĺpca v hlavičke tabuľky. Po vyplnení všetkých údajov môže byť tabuľka pridaná do databázy tlačidlom *Add Table*, alebo sa pridanie môže stornovať

tlačidlom *Cancel*. Pri vymazaní celého stĺpca z tabuľky môže dôjsť k tomu, že sa v tabuľke vyskytnú rovnaké riadky. V takomto prípade je pri pokuse pridať tabuľku ohlásená chyba a je nutné nadbytočné riadky vymazať.

4.5 Vyhodnotenie a krokovanie dotazu

Vyhodnotenie dotazu je možné spustiť tlačidlom *Evaluate query* z toolbaru, menu, alebo klávesou F5. Pri vyhodnocovaní sa aktivuje tlačidlo *Stop*, ktorým je možné proces stornovať. Začne prebiehať vyhodnocovanie a pri zložitejších dotazoch môže užívateľ sledovať postup práce v stavovom riadku. Po úspešnom ukončení vyhodnotenia sa zobrazí strom dotazu a aktivuje sa tlačidlo *Trace query*, ktorým sa spúšťa krokovanie dotazu. *Breakpointy* sa dajú na jednotlivé uzly stromu pridať ešte pred spustením krokovania.

Po spustení režimu krokovania nad vyhodnoteným dotazom sa automaticky skryjú nepotrebné okná dotazov a tabuliek, a pre každú premennú v dotaze sa otvorí plávajúce okno s tabuľkou(*Watch*), ktorej riadky sa do premennej budú dosadzovať. V pravej časti hlavného okna sa otvorí plávajúce okno *Variable watch* s aktuálnymi hodnotami všetkých premenných. V tomto okne je vľavo vedľa názvu premennej aj farba, ktorou je vyznačený aktuálny riadok v tabuľkách okien *Watch*.



Obr. 4.5 Okno aplikácie TRC – režim krokovania

Po spustení režimu krokovania sa proces zastaví aby sa užívateľ mohol v novom usporiadaní okien a premenných zorientovať. Krokovanie sa opäť spustí tlačidlom *Trace*, alebo klávesou F6, ktorej stlačenie spusí animáciu pri ktorej sa postupne označujú aktuálne hodnoty premenných v tabuľkách. Rýchlosť tejto animácie môže voliť užívateľ v nastaveniach aplikácie. Prednastavená hodnota rýchlosti animácie je 5 milisekúnd. Nastavením tejto hodnoty na 0 ms, sa animácia vypne a až do zastavenia na prvom breakpointe sa dotaz vyhodnocuje rovnako ako v režime samotného vyhodnocovania. Zmena tohto času sa prejaví až pri nasledujúcom spustení krokovania dotazu. Proces vyhodnotenia opakovane spúšťame klávesou F6 a pri nasledujúcom breakpointe sa vyhodnocovanie zastaví. Tento postup sa opakuje až do chvíle ked boli za premenné v dotaze dosadené všetky kombinácie riadkov tabuliek týchto premenných. Pri zastavení na breakpointe sa príslušný uzol označí žltou šípkou a následne užívateľ môže označovaním jednotlivých uzlov sledovať ich hodnoty, a výsledky operácií operačných uzlov.

Vyhodnocovanie prebieha od najnižších vrstiev stromu smerom hore, a vyhodnocuje sa vždy najprv ľavý podstrom operačného uzlu a potom pravý, preto je poradie zastavovania na breakpointoch umiestnených na jednej vrstve vždy zľava. Pri sledovaní hodnôt v strome sú aktuálne hodnoty uzlov len na hladine nižšej ako je hladina uzlu na ktorom je aktívny breakpoint(aktívny znamená, že proces zastal na tomto uzle), pretože vyhodnotenie dosiahlo len po tento uzol a do vyšších vrstiev prejde až po jeho vyhodnotení. Ak sa proces vyhodnotenia postupne dostane až ku koreňu stromu, podľa jeho hodnoty sa na výstup pridajú n-tice dosadené do voľných premenných. Akákoľvek zmena vo výslednej tabuľke sa užívateľovi zobrazí zvýraznením tlačidla *DataView* na hlavnom paneli. Ak sa pri vyhodnotení na výstup dostane viackrát rovnaký riadok, je do výsedku zahrnutý len jeho prvý výskyt a pokus o pridanie je vypísaný do okna *Output list*.

Po ukončení krokovania dotazu sa opäť zastaví celý proces a až po stlačení tlačidla F6 sa potvrdí zatvorenie tohto režimu a obnoví sa pôvodné nastavenie okien.

4.6 Nastavenia programu

Zmeniť vlastnosti užívateľského prostredia aplikácie a ďalších funkcií programu je možné vo formulári nastavení, ktorý sa objaví po zvolení položky *Options* z hlavného menu, alebo klávesovou skratkou Ctrl+P. Označením položky v ľavej časti formulára nastavení sa v pravej časti zobrazia objekty ktorých vlastnosti je možné meniť. Užívateľ môže definovať tieto vlastnosti:

Animation step – nastavuje rýchlosť animácie pri krokovaní dotazu tým, že sa určí čas, na ktorý sa animácia po vyznačení aktuálneho riadku premennej pozastaví. Tento čas sa určuje v milisekundách. Minimálna hodnota je 0 - animácia je vypnutá, maximálna hodnota je 2500.

Antialiasing – zapne alebo vypne vyhladzovanie obrazu pri vykresľovaní stromu dotazu.

Fonts – nastavenia písma používaného v programe.

- *Query text* – písmo, ktoré sa použije v okne Query input pre zadanie dotazu.
- *Query symbols* – písmo ktorým sa vykresľujú symboly operácií v okne pre zadanie dotazu. Neodporúča sa meniť toto nastavenie. Štandardný font použitý pre symboly je *Symbol.*
- *Tree text* – písmo ktoré sa použije na popis uzlov v strome.
- *Tree symbols* – písmo operácií na operačných uzloch v strome. Neodporúča sa meniť toto nastavenie. Štandardný font použitý pre symboly je *Symbol.*

Colors – nastavenia farieb stromu. Pre uzly sa nastavuje farba neoznačneného a farba označeného uzlu. Odporúča sa nastaviť odlišné farby aby sa dal označený uzol rozpoznať od ostatných. Nastavujú sa tieto farby:

- *Tree background* – farba pozadia stromu.
- *Node connection* – farba spojníc uzlov.
- *Query symbols* – farba symbolov logických operácií v okne *Query input*.
- *Operation node* – farby operačného uzlu.
- *Data node* – farby datového uzlu.
- *Math operation* – farby uzlov matematických operácií.
- *Predicate node* – farba operačného uzlu s operáciou predikátu.

Query tree – v tejto časti sa nastavujú rozmery uzlov a vzdialenosti v strome v pixeloch. Je možné nastaviť tieto hodnoty:

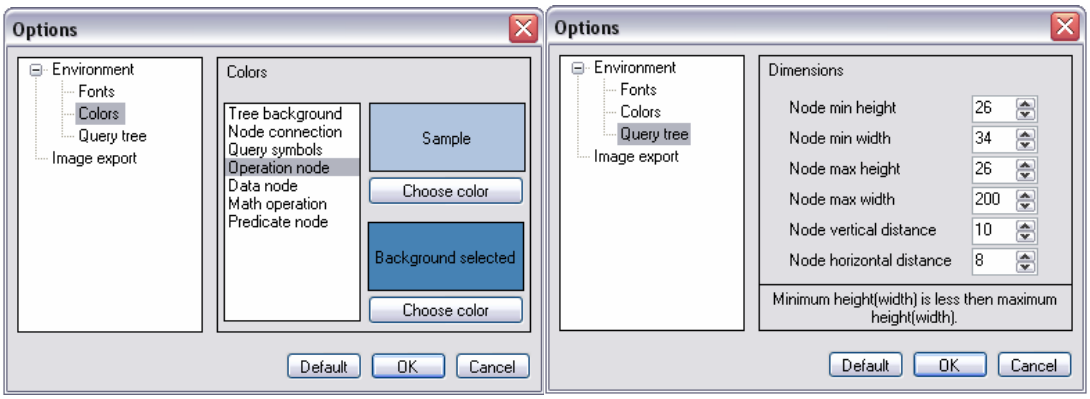
- *Node min height/width* – minimálna výška/šírka uzlu. Táto hodnota bude použitá, aj keď titulok uzlu nezaberie celú časť tela uzlu. Ak text na popis uzlu prekročí jeden z týchto rozmerov, uzol sa zväčší. Minimálna hodnota výšky je 24 px, šírky 34px.
- *Node max height/width* – maximálna výška/šírka uzlu. Rozmery titulku budú skrátené vždy na tieto hodnoty. Maximálna hodnota ktorú je možné nastaviť je 500px.
- *Node vertical/horizontal distance* – vertikálna/horizontálna vzdialenosť uzlov rozhoduje o celkovej výške alebo šírke stromu. Minimálna vzdialenosť je 10px, maximálna 50 px.

Pri nastavovaní rozmerov musí platiť, že hodnota maximálnej dĺžky/šírky uzlu je väčšia ako hodnota minimálnej dĺžky/šírky.

Image export – obsahuje nastavenia pre uloženie stromu dotazu do súboru. Obsahuje položky:

- *Query text location* – určí polohu textu na obrázku.
- *View query string* – umožní zobrazenie textu dotazu na obrázok.

Po nastavení vlastností sa nové nastavenie potvrdí stlačením tlačidla *OK* na formulári *Options*. Tlačidlo *Default* slúži na pôvodné nastavenie všetkých hodnôt.



Obr. 4.6 Okno nastavení aplikácie TRC

5 Štruktúra programu a problémy pri implementácii

Predpokladom výukovej aplikácie je vytvorenie jednoduchého a prehľadne členeného užívateľského rozhrania. Zadanie práce predpokladá tiež grafický výstup aplikácie, a preto bola aplikácia napísaná v jazyku C# s použitím .NET Frameworku, ktorého použtie omnoho uľahčilo vytvorenie užívateľského rozhrania k aplikácii a prácu s grafikou.

5.1 Implementácia užívateľského rozhrania

Plávajúce okná

Prehľadnosť v užívateľskom rozraní zvyšuje použitie plávajúcich okien, ktoré zároveň umožňujú užívateľovi prispôsobiť si pracovné prostredie. Na vytvorenie plávajúcich okien s funkciou prichytávania k okrajom hlavného okna bola v práci použitá knižnica *MagicLibrary.dll* [4], ktorej funkcie umožňujú jednoducho vytvoriť plávajúce okno obsahujúce ľubovoľný počet ďalších komponentov. Funkčnosť obsahu plávajúcich okien je oddelená od mechanizmu ich prichytávania a premiestňovania, ktoré má na starosti inštancia triedy *DockingManager* z knižnice [4]. Pred vytvorením okien je potrebné nadefinovať objekty v hlavnom okne, ku ktorým sa plávajúce okná môžu prichytávať, a naopak ku ktorým nesmú. V aplikácii TRC je možné prichytiť tieto okná k ľavému a pravému okraju hlavného okna, k hornému okraju stavového riadku a spodnému okraju hlavného toolbaru. Vymedziť objekty ku ktorým sa okná prichytávajú je možné nastavením *property InnerControl* a *OuterControl* triedy *DockingManager*. Okná je možné zoskupovať do kariet(*Contents*), ktoré zobrazia aktuálne len jedno z okien, ale umožňujú prepínanie medzi týmito oknami. Objekty sa do karty pridávajú metódou kolekcie kariet *DockingManager.Contents.Add*.

V projekte je z tejto knižnice použitá aj trieda *ResourceHelper* na načítanie ikon a kurzorov zo súboru a vytvorenie tzv. *resources*, ktoré možno použiť namiesto načítavania grafiky a kurzorov zo súboru pri každom spustení aplikácie.

Vizualizácia stromu dotazu

Základom vizualizácie dotazu je vykreslenie jeho stromu. Na kreslenie jednoduchých geometrických útvarov sú použité funkcie a triedy z namespace *System.Drawing*. Štruktúra použitá pre vyhodnotenie aj vykreslenie dotazu je binárny strom. Po postavení a vyhodnotení stromu je nutné najprv vypočítať súradnice a veľkosti jednotlivých uzlov tak, aby ich obsah nepresahoval telo uzlu, a boli dodržané vzdialenosti a rozmery uzlov definované v nastaveniach.

Na redukovanie šírky binárneho stromu bol použitý jednoduchý algoritmus, ktorý je súčasťou vyhodnocovania dotazu. Ten pri prechode uzlami zľava do prava postupne počíta dĺžku jednotlivých uzlov a tú pripočítava k aktuálnej horizontálnej vzdialenosti. Tým získa minimálnu x-ovú súradnicu na ktorú je možné umiestniť nasledujúci uzol na rovnakej hladine. Redukciou sa šírka stromu zmenší a zaplnia sa voľné miesta pre chýbajúce uzly v nevyváženom strome.

Nakreslenie stromu s vypočítanými súradnicami pre jednotlivé uzly prebehne rekurzívne. Najprv sa vykreslí spojnica uzlov, potom telo uzlu, ktoré je tvorené obdĺžnikom s oblými rohmi, a farbou podľa typu uzlu. Nakoniec sa vykreslí príslušným fontom popis uzlu a rekurzívne sa vykreslia ľavý a pravý podstrom. Kresliť grafiku je možné do objektu *Graphics*, ktorý môžeme získať od takmer ktoréhokoľvek komponentu okna metódou *Graphics.FromHwnd*, ktorá má jeden parameter, a tým je handle komponenty.

V prípade aplikácie TRC sa strom vykresľuje na komponent *System.Windows. Froms.Panel*. Spomínaný postup vykreslenia na okno ale nestačí. Pri pohybe oknom, alebo prekrytí okna aplikácie iným oknom, sa určitá oblasť okna zneplatní, to znamená že sa vyfarbí farbou pozadia komponentu a táto časť obrázku sa musí znovu prekresliť. Bez opakovaného prekresľovania by obrázok napr. po minimalizácii okna zmizol. Tento problém riešia všetky okenné systémy podobne. Pri zakrytí a následnom odkrytí časti okna iným objektom je tomuto oknu vyslaná správa o výskyte tejto udalosti spolu so súradnicami a rozmermi útvaru, ktorý treba prekresliť, a referenciou na inštanciu objektu *Graphics*, do ktorého sa má kresliť. V .NET Framework je možné preťažiť funkciu ktorá na túto správu reaguje. Táto funkcia sa často označuje ako *OnPaintEvent.* Pre správne prekresľovanie obrázku je teda potrebné umiestniť kód vykresľovania do tela metódy *OnPaint*, ktorej vstupný parameter obsahuje plochu, ktorú je treba prekresliť, a tá je definovaná ako reťazec obdĺžnikov.

Ďalší problém vznikne, ak sa bude vykresľovať stále celá plocha stromu, a nebude sa brať ohľad len na zneplatnenú oblasť okna. V takom prípade sa pri prekreslení najskôr vymaže a potom znovu nakreslí celý obrázok, čo je veľmi náročné. Pri pohybe okna sa môže *OnPaintEvent* vyvolať aj niekoľkokrát za sekundu, čo spôsobí preblikávanie obrazu. Na odstánenie tohto javu sú dve možnosti, prvou z nich je použitie vykresľovacej funkcie, ktorá kreslí len časť obrázku v neplatnej oblasti okna. Pretože sa v aplikácii TRC vykresľuje zložitejšia štruktúra stromu, bolo použité implementačne menej náročné riešenie s pomocou bufferu. Princíp spočíva v tom, že obrázok sa vykreslí len raz, a uloží sa do bufferu, ktorý má v aplikácii TRC podobu bitmapy *System.Drawing.Bitmap*. Po vyvolaní *OnPaintEvent* sa do panelu vyreslí metódou *Graphics.DrawImage* len obdĺžnik z obrázku uloženého v bitmape. Použitím tejto funkcie ktorá je oproti vykresleniu stromu rýchlejšia, a kopírovaním len potrebnej časti obrázku eliminujeme blikanie obrazu. Nevýhodou je vyššia pamäťová náročnosť programu, ktorý musí v pamäti udržiavať bitmapu celého stromu.

Ak rozmery vykresleného stromu presahujú rozmery hlavného panelu, sú automaticky zobrazené posuvné lišty, ktorými je možné posúvať obrázok tak, aby boli postupne viditeľné všetky časti stromu. Rovnakú funkciu má aj použitie prostredného tlačidla

myši. Na posun obrázku je použitý rovnaký systém ako na prekresľovanie. Pri pohybe posuvnej lišty sa sníma dĺžka posunutia v pixeloch, a do oblasti hlavného panelu sa skopíruje obrázok z bufferu s posunutím príslušnej súradnicovej osi o dĺžku posunutia lišty. Výsledný efekt je ten, že pri pohybe lišty, sa obraz hýbe v opačnom smere tohto pohybu. Rovnako sa rieši aj zmena veľkosti celého okna, keď sa prekresľuje len časť stromu pri zväčšujúcom sa okraji okna.

V aplikácii TRC je na vykresľovanie použitá trieda *DrawStruct* s položkami *Bitmap bBack* a *Graphics gBack.* Objekt *Graphics* získame priamo z bitmapy.

gBack = Graphics.FromImage(bBack);

Objekt *DrawStruct* teda obsahuje bitmapu, ktorá sa používa na samotný výstup na okno, a objekt *Graphics*, do ktorého sa kreslí, čím sa priamo mení bitmapa. Po akejkoľvek zmene v obrázku (napr. označenie uzlu = vykreslenie uzlu inou farbou), sa zmení bitmapa v štruktúre *DrawStruct*, ale nemusí sa okamžite vykresliť na panel. Pre okamžitú aktualizáciu je potrebné vyvolať *OnPaintEvent*, na čo použijeme metódu *Invalidate*.

GridWCaption

Zaujímavosťou užívateľského prostredia aplikácie TRC je použitie komponentov na farebné označovanie riadkov v tabuľkách pri krokovaní dotazu. Štandarndý prvok na zobrazovanie tabuliek v prostredí .NET je *DataGridView*, ten ale neumožňuje vyznačiť viacero riadkov súčasne, a ani zvoliť farbu vyznačenia. Ďalšou nevýhodou je, že oproti svojej staršej verzii (*DataGrid)*, neobsahuje titulkový riadok s názvom tabuľky. To viedlo k vytvoreniu nového komponentu *GridWCaption* ktorý vyhovuje požiadavkám aplikácie TRC. Nový komponent je potomkom triedy *DataGridView*, naviac však obsahuje titulkový pruh a umožňuje spomínané vyznačovanie riadkov. Vyznačenie riadkov nesúvisí s datovým obsahom tabuľky, označuje sa iba riadok na konkrétnej pozícii. Aby bolo možné v jednej tabuľke vyznačiť aktuálne riadky viacerých premenných, obsahuje *DataGridView* zoznam premenných, a ku každej premennej pozíciu aktuálneho riadku, ktorý má byť vyznačený, a farbu ktorou sa riadok vyfarbí. Na vyfarbenie konkrétnych riadkov jednotlivých premenných sa preťaží metóda *RowPrePaint*, ktorá sa volá vždy pred vyplnením obsahu konkrétneho riadku. Táto metóda zistí, či aktuálne prekresľovaný riadok nieje jeden z aktuálnych riadkov niektorej premennej, a ak je, vyberie sa farba premennej a pozadie riadku sa vymaľuje touto farbou. S každou premennou sa do zoznamu pridá aj farba, ktorú bude premenná používať. V aplikácii TRC je pre prvých 5 premenných už priradená farba. Ak sa v dotaze vyskytuje viac ako 5 premenných, ostatným premenným je vygenerovaná ďalšia náhodná farba. Po inicializácii komponentu je číslo riadku premenných nastavené na -1, a vyznačenie riadku sa nezobrazí. Rovnako pri použití čísla riadku ktorý sa v tabuľke nenachádza nedôjde k vyznačeniu. Preto je po zmene datového zdroja pre *DataGridView* celý zoznam premenných s pozíciami riadkov a farbami vymazaný. Metódou na vyznačovanie riadkov je *AssignVariablePos*, ktorej parametrami sú názov premennej a pozícia aktuálneho riadku. Táto funkcia zmení hodnotu aktuálneho riadku

pre danú premennú a vyvolá prekreslenie starého, aj novo-vyznačeného riadku metódou *Invalidate*. Funkcia na priradenie novej pozície danému riadku je prístupná aj ako *delegate* funkcia ktorú možno volať na objekt umiestnený v inom vlákne – to sa používa pri krokovaní dotazu.

5.2 Implementácia vyhodnocovania dotazu

Aby aplikácia mohla pracovať s tabuľkami a vyhodnocovať dotazy, potrebuje každá inštancia aplikácie prístup k databáze, ktorá slúži ako zdroj dát pre vyhodnocovanie. Ďalej je potrebná štruktúra v ktorej sa uchovávajú dotazy spolu so svojím popisom, množina nastavení pre vyhodnotenie dotazu, ale aj pre celkové fungovanie aplikácie, a nakoniec mechanizmus vyhodnocovania dotazu.

5.3 Databáza

Vnútornú databázu aplikácie tvorí inštancia vstavanej triedy *DataSet.* Túto databázu používa aplikácia pri vyhodnocovaní dotazu. Po otvorení súboru projektu sa všetky tabuľky projektu načítajú do *DataSet*-u aplikácie. Tento súbor tabuliek je možné ďalej rozšíriť vytvorením nových tabuliek, alebo načítaním tabuľky zo súboru. Naopak pri uložení projektu sa vyberú všetky tabuľky z *DataSet*-u a vložia sa do súboru. Obmedzenie každej tabuľky, ktoré zakazuje nulové bunky zabraňuje vzniku problému, ktorý by nastal pri porovnávaní hodnôt s prázdnymi bunkami, a zároveň umožňuje pri vyhodnocovaní použiť metódu *Contains*, ktorú obsahuje každá tabuľka na vyhľadanie riadku s danými hodnotami. Pre celý *DataSet* je štandardne definovaná serializácia a preto bolo možné bez problémov použiť triedu *BinaryFormatter* na uloženie tabuliek do binárneho súboru projektu. Tabuľky sa osobitne ukládajú do textového xml súboru metódou *DataSet.WriteXml.*

5.4 Dotazy a nastavenia

Dotaz v aplikácii TRC reprezentuje trieda *Query*, ktorá obsahuje okrem textu dotazu aj poznámku, titulok, a presne formátovaný text tak, ako ho zadal užívateľ v tvare *RTF*. Aktuálne uložené dotazy aplikácie obsahuje trieda *DataStore,* ktorá je podobne ako aktuálna databáza aplikácie načítaná zo súboru, a je možné ju dopĺňať o ďalšie dotazy.

Nastavenia aplikácie a rôzne pomocné konštanty a premenné sú uložené v inštancii triedy AppOptions. Trieda používa na uloženie rôznych nastavení štruktúru *Hashtable*,

ktorá môže obsahovať objekty rôzneho typu, a na tieto objekty sa možno odkazovať použitím identifikátora, pod ktorým sú objekty do *Hashtable* vložené. *AppOptions* obsahuje nastavenia rozčlenené v kategóriách:

Hashtable pens – obsahuje nastavenia pier používaných pri kreslení stromu.
Hashtable colors – farby uzlov stromu a pozadia stromu.
Hashtable brushes – inštancie tried Brush použitých pri vykresľovaní stromu.
Hashtable dimensions – rozmery uzlov v strome a rozmery hlavného okna.
Hashtable options – ostatné nastavenia, napr. rýchlosť animácie, antialiasing.

Prístup k týmto nastaveniam je prostredníctvom vlastností (*properties*) triedy *AppOptions*, ktoré umožňujú jednoduchý prístup ku konkrétnym údajom z *Hashtable.*

Aby sa nastavenia uchovali aj po vypnutí programu, zapisujú sa do pomocného súboru, z ktorého sú pri štarte aplikácie znovu načítané. Pri inštalácii sa do pomocného adresára pre aplikácie (Najčastejšie to je cesta *Documents and Settings\User\Application Data\TRC*) uloží aj súbor nastavení *opts.cfg.* Tento binárny súbor obsahuje inštanciu triedy *AppOptions* v prezistentnom stave. Na serializovanie triedy bol použitý *BinaryFormatter*, v triede bolo ale nutné implementovať interface *ISerializable*, ktorý umožní serializáciu do binárneho súboru previesť. Potrebný interface trieda spĺňa po pridaní špeciálneho konštruktoru

AppOptions(SerializationInfo info, StreamingContext ctxt)

ktorý inicializuje položky triedy *AppOptions* podľa obsahu triedy *SerializationInfo*, ktorá obsahuje údaje zo súboru. Opačný postup, teda uloženie do súboru, zabezpečuje metóda *void GetObjectData(SerializationInfo info, StreamingContext ctxt).* Ak sa pri spustení aplikácie súbor s nastaveniami nepodarí nájsť, použije sa na inicializáciu nastavení štandardný konštruktor.

5.5 Vyhodnocovanie dotazu

Jednou z dôležitých štruktúr aplikácie je strom dotazu reprezentovaný triedou *ExpTree* a jeho uzly, ktoré sú inštanciou triedy *ExpNode*. Strom nieje iba štruktúra na vyhodnocovanie, ale slúži aj ako predloha pre grafický výstup a preto obsahuje množstvo ďalších položiek ktoré sú potrebné pre vykreslenie stromu.

Uzol *ExpNode* je základným prvkom binárneho stromu, ktorý je definovaný rekurzívne, teda každý uzol si udržiava referenciu na svojho ľavého a pravého potomka. Strom je potom definovaný jediným uzlom – koreňom stromu. Medzi najdôležitejšie položky uzlu *ExpNode*, potrebné pre vyhodnotenie dotazu patria:

- *Data Type node_type* – určuje datový typ hodnoty uzlu. Podľa tejto položky sa rozpozná aké údaje uzol obsahuje, a pri operáciách s týmto uzlom sa najprv

skontroluje či je typ hodnoty uzlu kompatibilný s použitou operáciou. Rozlišujeme niekoľko dátových typov uzlov, ich zoznam a popis je uvedený v kapitole 3.2.

- *bool Bdata, double Ndata, string Sdata* – každý z datových typov uzlov špecifikovaných položkou node_type sa skonvertuje na jeden z troch štandardných datových typov. Položka Bdata je určená pre uzly typu *BOOL*, do Ndata sa skonvertujú hodnoty uzlov typu *NUMBER*, a *DATE*, a Sdata je určená pre uzly typu *STRING*, ktoré reprezentujú reťazcovú konštantu, ale aj *VARIABLE*, ktoré položku Sdata používajú na uloženie identifikátora premennej, názvu stĺpca alebo tabuľky.
- *List<KeyValuePair<String, int>> query* – časť dotazu, ktorú uzol reprezentuje je uložená ako zoznam párov *String, int*, kde každý pár predstavuje samostatný token – teda identifikátor premennej, číslo, operátor alebo zátvorka.
- *ExpNode left, right* – sú referencie na potomkov v strome, hodnota *null* týchto položiek znamená, že vetva stromu ďalej nepokračuje.

ExpNode je len abstraktnou triedou. V skutočnosti sú ale uzly rozdelené na datové a operačné. Operačný uzol reprezentuje trieda *OperNode*, a datový uzol trieda *DataNode*. Obe tieto triedy sú potomkom *ExpNode*, a na odlíšenie inštancií týchto tried sa používa funkcia *GetNodeType*, ktorá určí, či ide o operačný, alebo datový uzol. Každý uzol obsahuje metódu *Evaluate*, ktorá daný uzol vyhodnotí.

Okrem položiek potrebných na vyhodnotenie uzlu, obsahuje každý uzol tieto položky pre vykresľovanie uzlu v strome a krokovanie dotazu:

- *int X, Y* – súradnice uzlu vzhľadom ku koreňu stromu ktorý je začiatkom sústavy a má súradnice {0,0}.
- *int Width, Height* – šírka a výška tela uzlu, ktorá sa vypočíta podľa dĺžky jeho titulku.
- *bool selected* – Pri označení uzlov kliknutím na vykreslený strom je potrebné si zapamätať ktoré uzly boli vybrané a nato je určená táto položka.
- *right/left_visible* – položky určujú či je ľavá a pravá vetva pre daný uzol viditeľná. Používa sa pri kreslení stromu.
- *bool breakpoint* – Ak je hodnota tejto položky nastavená na *true*, je pre uzol nastavený breakpoint, a pri vykreslení sa uzol vyznačí šípkou. Pri krokovaní sa kontroluje táto položka, a ak je na danom uzle breakpoint, vyhodnotenie sa zastaví a uzol sa zvýrazní.

Hodnoty uzlov v strome je možné sledovať po kliknutí na daný uzol. Nato aby sme rozpoznali či bod so súradnicami kurzora myši leží v tele nejakého uzlu, alebo mimo neho sa používa funkcia *Hit*, ktorej parametrom sú súradnice bodu na ktorý ukazuje kurzor myši. Po kliknutí na panel sa najprv zistí, či bod patrí do oblasti stromu, a potom sa zavolá metóda *Hit* na koreň stromu, ktorá otestuje prienik tela uzlu a bodu, ak bod neleží v uzle, zavolá sa funkcia *Hit* pre ľavého alebo pravého potomka uzlu podľa polohy bodu na ktorý užívateľ klikol a súradnice uzlu.

Trieda *ExpTree* obsahuje referenciu na koreň stromu, ktorým je vždy operačný uzol. Vyhodnotenie a krokovanie dotazu zabezpečujú práve metódy *BuildAndEval* a *TraceEval* triedy *ExpTree*. Už z názvu týchto metód sa dá vyčítať, že proces vyhodnotenie pozostáva z dvoch hlavných častí, a to sú stavba stromu a následné vyhodnotenie stromu. Pre názornosť ukážeme postup vyhodnotenia na príklade. Nech je daný vstupný dotaz:

$$\{ film[MENO\_FILMU] \mid FILM(film) \wedge film.ROK=2001 \} \quad 5.8.1$$

A nech vyhodnotenie prebieha nad databázou ktorá obsahuje tabuľky na obr 2.1. Pri zavolaní konštruktoru na vstupný dotaz v textovej podobe sa rozdelí text na jednotlivé tokeny metódou *SplitTRCQuery* triedy *Parser*, ktorá sa používa na prácu s textom dotazu. Tokenom označme najmenšiu časť dotazu ktorá má samostatný význam, teda token môže byť identifikátor, operátor alebo zátvorka. Ďalej sú odstranené medzery a všetky biele znaky medzi tokenmi, a je vytvorený dotaz v tvare zoznamu párov:

$$List<KeyValuePair<String, int>> query \quad 5.8.2$$

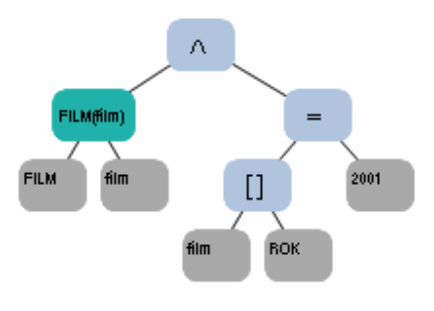
Každý pár reprezentuje jeden token, a obsahuje okrem jeho hodnoty aj pozíciu na ktorej text tokenu vo vstupnom formátovanom texte začína. Táto pozícia sa použije pri vyznačení prípadnej chyby pri vyhodnotení vo vstupnom okne pre zadávanie dotazu. Pri vyhodnotení algoritmus predpokladá použitie kľúčových slov pre logické operácie, a preto sú v zozname(5.8.2) nahradené znaky logických operátorov za kľúčové slová. V konštruktore sa ešte rozdelí dotaz na ľavú a pravú stranu, skontroluje sa správnosť syntaxe ľavej strany dotazu, a vytvorí sa zoznam voľných premenných. Po konštrukcii stromu *ExpTree* sú správne vyplnené jeho položky:

- *List<String> var* – voľné premenné dotazu, premenné z ktorých sa vytvorí výsledná relácia. Pre dotaz 5.8.1 by tento zoznam obsahoval len jedinú premennú *film.*
- *List<List<String>> col* – pre každú voľnú premennú sa zaznamená zoznam stĺpcov z ktorých sa má vytvoriť výsledok. Pre dotaz 5.8.1 bude táto položka obsahovať jeden zoznam s jedniným prvkom a tým je *MENO_FILMU.*
- *List<KeyValuePair<String, int>> right_side* – obsahuje hlavnú formulu dotazu uloženú ako zoznam tokenov. Pre dotaz 5.8.1 je tento zoznam vytvorený z tokenov *FILM, (, film,), ∧, film, ., ROK, =, 2001*.

Po konštrukcii stromu sa môže zavolať metóda *ExpTreeBuild*, ktorá zo zoznamu tokenov postaví strom. Je to funkcia, ktorá sa volá rekurzívne, jej parametrom je formula dotazu v tvare zoznamu tokenov, a výstupom je uzol *ExpNode*. Princíp parsovania zoznamu tokenov je taký, že *ExpTreeBuild* sa pokúsi nájsť operátor v tomto zozname, a ak sa to podarí, vytvorí sa operačný uzol konkrétneho typu a jeho operandy sú výsledky rekurzívne zavolanej funkcie na ľavú a pravú časť vstupnej formule (časti naľavo a napravo od nájdeného tokenu operátora). Ak sa operátor vo vstupnom zozname

tokenov nepodarí nájsť, výsledkom *ExpTreeBuild* je datový uzol. Takýmto spôsobom sa vytvorí strom dotazu, ktorého koreň sa uloží do položky root stromu *ExpTree*.

Vytvorené uzly majú svoj typ a pre operačné uzly je známy aj typ operácie. Hodnota uzlu a jej datový typ je ale zatiaľ neznámy. Uzly majú vyplnenú iba položku *query*, teda zoznam tokenov pre časť dotazu ktorú reprezentujú. Vytvorený strom dotazu 5.8.1 má tento tvar:



Pred vyhodnocovaním je nutné vedieť relácie všetkých premenných v dotaze, aby bolo možné n-tice týchto relácií dosadzovať do premenných. Určovanie aktuálnych hodnôt premenných a správa ich relácií je úlohou triedy *DomainTable*. Najdôležitejšími položkami triedy *DomainTable* je zoznam premenných, a pre každú premennú zoznam relácií. Premennými sa do táto tabuľka vyplní už pri stavbe stromu. Pridajú sa najprv všetky voľné premenné z ľavej strany dotazu, a potom premenné viazané v niektorej z podformúl pravej strany dotazu. Pre každú premennú v musí ale v dotaze existovať aspoň jedna podformula tvaru *Rel(v)*. Pri nájdení podformuly tvaru predikátu v dotaze, sa relácia *Rel* pridá do zoznamu relácií v triede *DomainTable*. Pretože je *DomainTable* prepojená priamo s hlavným *DataSet*-om aplikácie, je možné skontrolovať či sú názvy relácií platné pre aktuálny *DataSet*. Ak je definovaných viacero relácií pre jednu premennú, prevedie sa kontrola zhodnosti schém relácií, ktorá spočíva v tom, že sa pre každý stĺpec tabuliek skontroluje zhodnosť typu a názvu stĺpcov. Pre príklad dotazu 5.8.1 by inštancia triedy *DomainTable* obsahovala jedinú premennú film, a jej jedinú reláciu FILM.

Postavený strom, s inštanciou *DomainTable*, kde je pre každú premennú známa aspoň jedna relácia, sa môže začať vyhodnocovať. Dosadzovacím mechanizmom vyhodnocovania je metóda *IterArrayStep*, ktorej parametrom je pole iterátorov cez riadky relácií každej premennej. Táto funkcia postupne dosadzuje do všetkých voľných premennných všetky možné kombinácie n-tíc z ich relácií, a pre každú kombináciu spustí vyhodnotenie stromu volaním rekurzívnej metódy *Evaluate* koreňového uzlu stromu. Z kombinácie n–tíc ,pre ktoré metóda *Evaluate* uzlu *ExpTree.root* je rovná true, sa zostaví výsledok.

Metóda *OperNode.Evaluate* vyhodnotí najskôr svojho ľavého a potom pravého potomka, a až po vyhodnotení obidvoch operandov sa určí hodnota uzlu. Pre datové uzly metóda *Evaluate* znamená určenie typu a hodnoty uzlu pomocou zoznamu tokenov,

ktorý bol vyplnený pri stavaní stromu. To znamená rozpoznať, či ide o identifikátor, číslo, alebo reťazec, a potom jeho hodnotu uložiť do príslušnej datovej položky uzlu (*Ndata, Sdata, Bdata*). Vyhodnocovanie datových uzlov v strome nieje nutné pri každom dosadení do premenných pretože ich vyhodnotenie je stále rovnaké, a preto sa pri prvom vyhodnotení nastaví zarážka, ktorá zabezpečí aby sa uzly pri dalšom prechode znovu nevyhodnocovali.

Zložitejším je len vyhodnotenie operačných uzlov typu kvantifikácie, ktoré je podobné systému celkového vyhodnocovania. Z *DomainTable* sa získa iterátor n-tíc pre premennú ktorá je kvantifikovaná, a postupne sa dosadzujú do tejto premennej n-tice z jej relácií a pre každé dosadenie sa spustí vyhodnotenie uzlu, ktorý je koreňom kvantifikovanej formuly. Rozdiel v typoch kvantifikátorov je ten, že pri existenčnom kv. hľadáme aspoň jedno dosadenie také aby formula platila, a pri všeobecnom hľadáme dosadenie, pre ktoré formula neplatí. Po nájdení takej n-tice je výsledok operácie s existenčným kv. pravda, a pre všeobecný kv. nepravda.

V prostredí .NET, v ktorom je aplikácia napísaná, sa kód každého okna alebo formulára vykonáva v osobitnom vlákne. Vyhodnocovanie výrazu je spúšťané tlačidlom hlavného okna, a to znamená že by proces vyhodnotenia bežal v rovnakom vlákne ako kód hlavného okna. Pre zložité dotazy, alebo dotazy nad rozsiahlou databázou by vyhodnotenie mohlo trvať dlhšie a to by spôsbilo zamrznutie hlavného okna, ktorého kód by sa prestal vykonávať. Riešením tohto problému v aplikácii TRC je spustenie vyhodnocovacej funkcie stromu v novom vlákne. Využívajú sa pritom štandardné prostriedky .NETu na prácu s vláknami. Nové vlákno – inštanciu triedy *Thread* vytvoríme pomocou konštruktoru, ktorého parametrom je procedúra ktorej kód sa má vykonávať v novom vlákne:

vEvalThread = new Thread(new ThreadStart(vQueryTree.BuildAndEval));

Vyhodnocovanie v novom vlákne beží mimo kód užívateľského rozhrania, ale odlišnosťou je, že akékoľvek výstupy z vlákna vyhodnocovania nemožno do vlákna hlavného okna posielať priamo (prostredníctvom referencií), ale sú nato určené tzv. delegáty. Delegát v c# je objekt podobný pointru na funkciu v C++, a umožňuje uloženie referencie na funkciu, a neskôr vykonanie jej kódu pomocou metódy *Invoke* daného delegátu. Pri deklarácii delegátu určujeme typ funkcie, teda návratovú hodnotu, a typ jej parametrov, pri definícii špecifikujeme ktorú funkciu chceme zavolať. V aplikácii TRC sa teda informácie z vlákna vyhodnocovania dostanú do vlákna hlavného okna vyvolaním delegátu hlavného okna metódou *Invoke* tohto delegátu, a parametre delegátu predávame ako pole objektov.

Príkladom použitia delegátu je priebežné doplnenie výstupnej tabuľky z vlákna vyhodnotenia do *DataSet*-u hlavného okna. Hlavné okno obsahuje funkciu:

fAssignTbl(DataTable t),

ktorá rozšíri výsledok o riadky tabuľky *t*, predanej ako parameter.

Hlavné okno obsahuje inštanciu delegáta:

dAssignTbl = new AssignTblDelegate(this.fAssignTbl),

ktorý je potom z vlákna vyhodnocovania volaný *Invoke* takto:

this.main_form.Invoke(main_form.dAssignTbl, new Object[] { table }),

kde *main_form* je referencia na hlavné okno aplikácie, a *table* tabuľka *DataTable* ktorú chceme poslať na výstup.

Vytvorenie nového vlákna na vyhodnocovanie dotazu bolo použité aj pri krokovaní procesu vyhodnotenia. Pri vytvorení nového vlákna môžeme proces vykonania tohto vlákna pozastaviť bez toho aby to negatívne vplývalo na užívateľské rozhranie. Na krokovanie vyhodnotenia dotazu je použitá rovnaká funkcia ako na samotné vyhodnotenie až nato, že ak pri krokovaní vyhodnotenia táto funkcia narazí na uzol ktorý má nastavený breakpoint, musí sa pozastaviť kód vyhodnocovania, a ja treba aktualizovať hodnoty premenných a uzlov v strome a potom tieto hodnoty poslať pomocou delegátov hlavnému oknu, ktoré sa postará o sprostredkovanie informácií užívateľovi. Na detekciu pozastavnia vlákna sa používa položka *bool Suspended* stromu *ExpTree*, a na samotné pozastavenie vykonávania kódu sú v aplikácii použité metódy *Monitor.Wait*, ktorá pozastaví vykonávanie kódu a metóda *Monitor.Pulse* ktorá prebudí pozastavené vlákno.

Chyby ktoré môžu nastať pri vyhodnocovaní dotazu majú rôzny charakter. Najčastejšie sú to syntaktické chyby, ale môžu to byť aj chyby spôsobené nedostupnosťou niektorých objektov potrebných pri vyhodnocovaní (napr. chyba databázy, alebo nedostupnosť tabuliek na ktoré sa odkazuje dotaz). V aplikácii TRC pri vyhodnocovaní neexistuje jednotné miesto kde prebieha kontrola chýb, pretože by osobitná kontrola bola príliš náročná. Chyby sa hlásia počas celého procesu vyhodnocovania až vo chvíli, keď pre nejakú chybu vyhodnocovanie nemôže pokračovať. Nevýhodou takého typu ohlasovania chýb je, že je ohlásená len prvá chyba a prípadný výskyt ostatných chýb sa užívateľ dozvie až po opravení ohlásenej chyby a znovuspustení vyhodnotenia. Keďže vyhodnocovanie beží v novom vlákne a kôli chybe nemôže ďalej pokračovať, je hlásenie zabezpečené triedou *ApplicationErrors* ktorá používa na ohlásenie chyby výnimky *ExprEvaluationException*. Ak charakter chyby umožňuje špecifikovať pozíciu chyby vo vstupnom texte dotazu potom je chyba vyznačená priamo vo vstupnom okne dotazu *Query input*. Ak sa nejedná o chyby syntaxe sú oznámené aspoň druhy chýb.

6 Záver

Cieľom tejto práce bolo vytvoriť program, ktorý vyhodnocuje a vysvetľuje princíp vyhodnotenia dotazov v NRK. Aplikácia TRC, ktorá je výsledkom práce, umožňuje vytvorenie jednoduchej databázy tabuliek, vyhodnotenie a vizualizáciu dotazu, a krokovanie procesu vyhodnotenia. Okrem toho aplikácia poskytuje prehľadné užívateľké prostredie ktoré je dôležité pre všetky aplikácie určené na výukové účely.

Výukovou aplikáciou s podobným zameraním ako je TRC, je program WinRDBI (Windows Relational DataBase Interpreter) [5]. Program je voľne šíriteľný a umožňuje vyhodnocovať dotazy nad relačnou databázou. WinRDBI je napísaný v Jave a určený pre operačný systém windows. Umožňuje vytvorenie jednoduchej relačnej databázy a vyhodnotenie dotazov v n-ticovom a doménovom relačnom kalkule, v relačnej algebre, ale aj v SQL. Celá aplikácia poskytuje jednoduché a zrozumiteľné užívateľské rozhranie s hlavným MDI oknom. Užívateľ môže databázu a dotazy osobitne ukládať do súborov.

Program WinRDBI je určený len na vyhodnocovanie dotazov a neposkytuje žiadne informácie o spôsobe vyhodnocovania. Značenie dotazov použité v tomto programe je štandardné a od aplikácie TRC sa líši len značením projekcie voľných premenných na ľavej strane dotazu. Rozdielom je tiež použitie veľkých písmen pre značenie premenných a malých písmen na značenie relácií, zatiaľ čo pri aplikácii TRC je značenie opačné. Program WinRDBI dokáže rozpoznať formuly ktoré niesú bezpečné a hlásenie chýb je porovnateľné s hlásením v programe TRC.

V budúcnosti by program TRC mohol byť rozšírený o ďalšie funkcie užívateľského rozhrania, ako aj o funkcie vyhodnotenia:

- Možnosť uložiť výsledok relácie do osobitnej tabuľky, ktorá by mohla byť použitá ako vstupná relácia pre iný dotaz.
- Práca s viacerými projektami naraz, alebo prenos informácií medzi inštanciami aplikácie TRC pomocou kopírovania do schránky windows.
- Zobrazenie hodnôt uzlov stromu pomocou farieb.
- Načítavanie tabuliek zo súborov iných aplikácií, napr. súbory *.xls MS Excel a iné.

Literatúra

[1] Codd E.F.: *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM 13 (6): 377–387, 1970.

[2] Ramakrishnan R., Gehrke J.: *Database management systems*, McGraw-Hill, 2003.

[3] Halška I., Pokorný J.: *Databázové systémy*, Vydavatelství ČVUT, 2003.

[4] MagicLibrary.dll, http://www.dotnetmagic.com/downloads/MagicInstall174.msi.

[5] The WinRDBI educational tool, http://www.eas.asu.edu/~winrdbi/

[6] Skopal T.: Prezentácia k prednáške Databázové systémy, http://siret.ms.mff.cuni.cz/skopal/databaze/slidy/DBI025_06.ppt