Prohlašuji, že diplomovou práci jsem vypracoval samostatně a použil jen prameny, které uvádím v seznamu použité literatury.

V Praze 24. května 2002 ……..

Abstrakt

Tato práce se zabývá rozborem procesů pro vstupy a výstupy z databáze dopravních dat, možností programových prostředků pro automatizované zpracování dat a příklady aplikací některých metod na reálná dopravní data s ohledem na použitelnost v uceleném dopravním informačním systému. Tento systém se skládá z databáze dopravních dat a jejího ovládacího programu, klasifikačního modulu, predikčního modulu, webového uživatelského rozhraní případně jiných zobrazovacích aplikací. Problémy klasifikace se ve své diplomové práci zabývá Michal Holík, predikcí Filip Škorpík a webové rozhraní v rámci své diplomové práce zpracoval Ing. Martin Stareček. Na tento systém volně navazují další aplikace jako například Off-line aplikace Ing. Otto Šenka a jiné. Systém by měl být přístupný i jiným modulům zabývajícími se například optimalizací řízení dané dopravní oblasti a podobně. Cílem tohoto projektu je poskytnout uživatelům internetu a nejen jim rychlý přehled o dopravní situaci a jejím vývoji v dané lokalitě.

Jako datový sklad slouží databázový systém Microsoft Access 2000 a její ovládací program je zpracován ve vývojovém prostředí Visual Basic 6 Enterprise firmy Microsoft. Predikční modul je založen na bázi umělých neuronových sítí. K jejich návrhu byl použit program NeuroSolution firmy NeuroDimension. Pro trénování a ověření činnosti byla použita reálná dopravní data naměřená v období prosinec 2001 a únor 2002. Klasifikační modul je zpracován v prostředí Visual Basicu 6 Enterprise a ke klasifikaci využívá shlukové analýzy. Pro vývoj webového rozhraní byl použit hypertextový preprocesor PHP ve spolupráci s interaktivní animací typu FLASH. Jako webový server pro lokální odlaďování byl použit server Apache.

Na tomto místě bych chtěl poděkovat především vedoucímu naší laboratoře, panu Doc. Ing. Mirko Novákovi, Drsc., za množství energie a podporu celého kolektivu, Ing. Emilu Pelikánovi, Csc., za poskytnuté odborné rady v oblasti predikcí časových řad, samozřejmě přátelům a spolutvůrcům celého tohoto systému. Dále Ing. Pavlu Hrubešovi, Ing. Pavlu Ptákovi a Ing. Tomášovi Chrpovi za podporu a pomoc. V neposlední řadě také mým rodičům za to, že mi umožnili studium na této škole.

Obsah

Předmluva

K výběru tohoto tématu pro diplomovou práci mě vedl zájem o moderní informační technologie z oblasti databázových systémů a jejich využití v problematice dopravy. V rámci projektově orientovaného studia na Fakultě dopravní jsem se zařadil říjnu 1998 v do projektu „ Využití umělých neuronových sítí pro detekci dopravních nehod “, kde jsem měl možnost podílet se na výzkumných pracích v oblasti slovní báze pro rozpoznávání hlasu a na výzkumu spolehlivosti interakce člověk - dopravní systém.

1. Databáze obecně

Pojem databáze dnes není zcela jistě nikomu cizí. Lidé mají potřebu evidovat a shromažďovat informace už odpradávna. Celá dnešní moderní společnost je postavena na databázových systémech, od evidence občanů, přes zdravotnictví, hospodářství, školství, až po letectví, výzkum nebo síť mobilních telefonů.

1.1 Vývoj databází

Nyní se vraťme asi o 40 let zpět. Právě v 60. letech minulého století vzniká současný pojem databáze, entita, atribut entity a vazba mezi entitami. To jsou základní pojmy, o kterých se zmíním.

Databázi si lze představit jako soubor dat, který slouží pro popis reálného světa (např. evidence školní knihovny, sklad chemikálií, evidence studentů). Entitou rozumíme prvek reálného světa (např. člověk, stroj, vyučovaný předmět, město), který je popsán svými charakteristikami (vlastnostmi). Ty se většinou označují jako atributy (např. jméno, příjmení, stav, plat, hmotnost).

Dalším důležitým pojmem je vazba mezi entitami. Jednotlivé entity odpovídající prvkům z reálného světa, mají mezi sebou určitý vztah. Např. každý člověk má právě jedny osobní údaje vedené na magistrátu, na oddělení občanských průkazů. To hovoříme o vazbě typu 1:1. Dalším typem je vazba 1:N, jíž bude odpovídat např. skutečnost, že jeden člověk může vlastnit více kreditních karet (ale jedna kreditní karta může být vlastněna pouze jedním člověkem). Posledním typem vazby je M:N. Zde není žádné omezení, příkladem by mohla být situace, že student na vysoké škole si může zapsat několik různých předmětů, ale jeden předmět může být zároveň zapsán více studenty.

1.2 Databázový model

V této době vzniká ještě jeden pojem, a to databázový model. Ten byl zaveden zejména matematiky, jako prostředek pro popis databáze. Zpočátku se používaly modely dvojího typu: hierarchický (založen na modelování hierarchie mezi

entitami se vztahy podřízenosti a nadřízenosti) a dále síťový (vychází z teorie grafů, uzly v grafu odpovídají entitám a orientované hrany definují vztahy mezi entitami). V 70. letech se uvedené databázové modely ukázaly být nedostatečné (objevily se problémy s realizací a implementací vazby M:N), a proto vznikl relační model, který se stal standardem a používá se dodnes.

1.2.1 Relační databáze

Základním pojmem je relace. Relaci, aniž bych zaváděl jakékoliv matematické definice, si lze představit jako tabulku, která se skládá ze sloupců a řádků. Sloupce odpovídají jednotlivým vlastnostem (atributům) entity. Údaje v jednom řádku tabulky zobrazují aktuální stav světa. Budu mít např. tabulku ZAMĚSTNANEC, která bude popisovat entitu pracovníka ve firmě. Sloupce tabulky budou: ČÍSLO, JMÉNO, PŘÍJMENÍ, DAT_NAR, PLAT a SMLOUVA_OD. Atribut DAT_NAR značí datum narození pracovníka a SMLOUVA_OD uvádí datum, od kterého je pracovník v naší firmě zaměstnán. Představme si, že reálnému světu odpovídá následující naplnění tabulky:

číslo	Jméno	přijemní	dat_nar	plat	smlouva_od
1	Jan	Novák	15.10.1975	15000	1.1.2000
2	Petr	Nový	1.4.1978	21500	12.5.1999
3	Jan	Nováček	6.9.1965	17500	7.7.1998

Tabulka je základním stavebním kamenem pro budování celé databáze. Relace tedy odpovídá celé tabulce a prvku relace odpovídá jeden konkrétní řádek. Jeden řádek bývá často nazýván databázovým záznamem. Soubor tabulek (relací) pak tvoří celou databázi (relační schéma).

U tabulek ještě chvíli zůstaňme. Jedna tabulka nám popisuje nějakou entitu. Za sloupce v tabulce zvolíme ty atributy, které o dané entitě chceme evidovat a které nás zajímají. Nyní je nejvyšší čas říci si něco o tvorbě relačních tabulek. Tvorba "dobře navržených" tabulek je klíčový a důležitý úkol zejména z hlediska dlouhodobého. (Pokud bychom v nějakém databázovém systému, který již nějakou dobu běží v ostrém provozu, nalezli nějakou chybu a zjistili bychom, že spočívá ve

špatně navržené databázi, mělo by to katastrofální důsledky, a to nejen po finanční stránce.) Dále vysvětlím další základní pojmy z oblasti relační databáze.

1.3 Základní pojmy

Hodnotou většinou rozumíme uživatelská data. Každý sloupec v tabulce má svůj datový typ (např. celé číslo, řetězec, datum, logická hodnota, apod.).

Pro práci s databázovými tabulkami je užitečné (ne-li přímo nutné) mít alespoň jednu položku (sloupec), jejíž hodnota nám bude jednoznačně identifikovat záznam v tabulce. Pokud taková položka nebude příliš velká (např. v počtu bajtů), zvolíme ji za tzv. primární klíč. V našem příkladu tabulky ZAMĚSTNANEC lze za primární klíč zvolit položku ČÍSLO. Primární klíč má tu vlastnost, že jeho hodnota je jedinečná, tj. pro žádné dva řádky v tabulce nemůže nastat situace, že by hodnota primárního klíče byla totožná. Databázové systémy většinou umožňují definovat jako primární klíč n-tici položek, např. dvojici nebo trojici položek. V takovém případě se mohou některé položky v klíčích opakovat, ale nesmí být shodné všechny položky dvou primárních klíčů najednou.

Neméně důležitá je i funkční závislost. Funkční závislosti si lze představit jako tvrzení o reálném světě. Například plat zaměstnance závisí na tom, jakou vykonává funkci, tj. plat závisí na funkci, zapisujeme FUNKCE->PLAT. Druhým příkladem by mohla být výše jízdného, která závisí na délce vlakové trasy, tj. DÉLKA_TRASY->VÝŠE_JÍZDNÉHO. Takových příkladů z běžného života bychom našli mnoho.

1.4 Normální formy

Pojem normálních forem se používá ve spojitosti s dobře navrženými tabulkami. Správně vytvořené tabulky splňují 4 základní normální formy.

1.4.1 První normální forma

První, nejjednodušší, normální forma (značíme 1NF) říká, že všechny atributy jsou atomické, tj. dále již nedělitelné (jinými slovy, hodnotou nesmí být relace). Mějme např. tabulku ADRESA, která bude mít sloupce JMÉNO, PŘÍJMENÍ a BYDLIŠTĚ. Naplnění tabulky nechť odpovídá reálnému světu:

jméno	přijemní	bydliště
Jan	Novák	Ostravská 16, Praha 16000
Petr	Nový	Svitavská 8, Brno 61400
Jan	Nováček	Na bradlech 1147, Ostrava 79002

Pokud bychom v této tabulce chtěli vypsat všechny pracovníky, jejichž PSČ je rovno určité hodnotě, dostali bychom se do potíží, neboť bychom to nemohli zjistit přímo a jednoduše. A to proto, že atribut BYDLIŠTĚ není atomický, skládá se z několika částí: ULICE, ČÍSLO, MĚSTO a PSČ. Správný návrh tabulky, který bude respektovat 1NF bude vypadat následovně:

jméno	přijemní	ulice	číslo	město	PSČ
Jan	Novák	ostravská	16	Praha	16000
Petr	Nový	svitavská	8	Brno	61400
Jan	Nováček	Na bradlech	1147	Ostrava	79002

Obecně bychom se měli snažit, aby obsahem jedné databázové položky byla právě jedna hodnota (určitého databázového typu).

1.4.2 Druhá normální forma

Tabulka splňuje 2NF, právě když splňuje 1NF a navíc každý atribut, který není primárním klíčem je na primárním klíči úplně závislý. To znamená, že se nesmí v řádku tabulky objevit položka, která by byla závislá jen na části primárního klíče. Z definice vyplývá, že problém 2NF se týká jenom tabulek, kde volíme za primární klíč více položek než jednu. Jinými slovy, pokud má tabulka jako primární klíč jenom jeden sloupec, pak 2NF je splněna triviálně. Nechť máme tabulku PRACOVNÍK, která bude vypadat následovně: (atribut ČÍS_PRAC značí číslo pracoviště, kde daný

pracovník pracuje, atribut NÁZEV_PRAC uvádí jméno daného pracoviště)

číslo	jméno	přijemní	čís_prac	název_prac
1	Jan	Novák	10	studovna
2	Petr	Nový	15	centrála
3	Jan	Nováček	10	studovna

Jaký primární klíč zvolíme v této tabulce? Pokud zvolíme pouze ČÍSLO, je to špatně, neboť zcela určitě název pracoviště, kde zaměstnanec pracuje, není závislý na číslu pracovníka. Takže za primární klíč musíme vzít dvojici (ČÍSLO,ČIS_PRAC). Tím nám ovšem vznikl nový problém. Položky JMÉNO, PŘÍJMENÍ a NÁZEV_PRAC nejsou úplně závislé na dvojici zvoleného primární klíče. Ať tedy děláme, co děláme, nejsme schopni vybrat takový primární klíč, aby tabulka splňovala 2NF. Jak z tohoto problému ven? Obecně převedení do tabulky, která již bude splňovat 2NF, znamená rozpad na dvě a více tabulek, kde každá už bude splňovat 2NF. Takovému "rozpadu" na více tabulek se odborně říká dekompozice relačního schématu. Správně navržené tabulky splňující 2NF budou vypadat takto: (tabulka PRACOVNÍK a PRACOVIŠTĚ)

číslo	jméno	přijemní	čís_prac
1	Jan	Novák	10
2	Petr	Nový	15
3	Jan	Nováček	10

číslo	název
10	studovna
15	centrála

Dále si všimněte, že pokud tabulka nesplňuje 2NF, dochází často k redundanci. Konkrétně v původní tabulce informace, že pracoviště číslo 10 se jmenuje "studovna", byla obsažena celkem dvakrát. Redundance je jev, který obvykle nesplnění 2NF doprovází. O tom, že redundance je nežadoucí, netřeba pochybovat

1.4.3 Třetí normální forma

Relační tabulky splňují třetí normální formu (3NF), jestliže splňují 2NF a žádný atribut, který není primárním klíčem, není tranzitivně závislý na žádném klíči. Nejlépe to opět vysvětlí následující příklad. Mějme tabulku PLATY, která bude vypadat takto:

Číslo	jméno	přijemní	funkce	plat
1	Jan	Novák	technik	15000
2	Petr	Nový	vedoucí	21500
3	Jan	Nováček	správce	17500

Pomineme zatím fakt, že tato tabulka nesplňuje ani 2NF, což je základní předpoklad pro 3NF. Chci zde jen vysvětlit pojem tranzitivní závislosti. Nebudeme přemýšlet, co je primární klíč, na první pohled vidíme, že konkrétně atributy JMÉNO, PŘÍJMENÍ a FUNKCE závisí na atributu ČÍSLO (ten by nejspíš byl primárním klíčem). Dále můžeme vidět, že atribut PLAT zřejmě je funkčně závislý na atributu FUNKCE a pokud vezmeme v úvahu, že ČÍSLO->FUNKCE a FUNKCE->PLAT, dostaneme díky tranzitivitě, že ČÍSLO->PLAT. Postup, jak dostat tabulky do 3NF je podobný jako v případě 2NF, tj. opět provedeme dekompozici: (tabulka FUNKCE a PLATY)

číslo	Jméno	přijemní	funkce
1	Jan	Novák	technik
2	Petr	Nový	vedoucí
3	Jan	Nováček	správce

funkce	plat
technik	15000
vedoucí	21500
správce	17500

Z hlediska základních tří normálních forem, jsou tyto dvě tabulky již v pořádku. Z praktického hlediska je vhodnější použít nějaký číselník funkcí, abychom splnili podmínku, že primární klíč v tabulkách má být co nejkratší délky. Nejlepší zápis je tedy následující:

číslo	jméno	přijemní	cis_fun
1	Jan	Novák	127
2	Petr	Nový	121
3	Jan	Nováček	156

cis_fun	funkce	plat
127	technik	15000
121	vedoucí	21500
156	správce	17500

1.4.4 Boyce-Coddova normální forma

Poslední formou je tzv. Boyce-Coddova normální forma (BCNF). Tabulka splňuje BCNF, právě když pro dvě množiny atributů A a B platí: A->B a současně B není podmnožinou A, pak množina A obsahuje primární klíč tabulky. Tato forma zjednodušuje práci s tabulkami, ve většině případů, pokud dobře postupujeme při tvorbě tabulek, aby splňovaly postupně 1NF, 2NF a 3NF, forma BCNF je splněna.

1.5 Jazyk SQL

Historie jazyka SQL spadá do 70. a 80. let. První standard byl přijat v roce 1986 (označován jako SQL86). Časem se však projevily některé nedostatky. Opravená verze je z roku 1992 a je označována jako SQL92. Ten je v oblasti relačních databází standardem dodnes. Zkratka SQL značí Structured Query Language. Jazyk v sobě zahrnuje nástroje pro tvorbu databází (tabulek) a dále nástroje na manipulaci s daty (vkládání dat, aktualizace, mazání a vyhledávání informací).

SQL patří do kategorie tzv. deklarativních programovacích jazyků, což v praxi znamená, že kód jazyka SQL nepíšeme v žádném samostatném programu (jako by tomu bylo např. u jazyka C, nebo Pascal), ale vkládáme jej do jiného programovacího jazyka, který je již procedurální. Se samotným jazykem SQL můžeme pracovat pouze v případě, že se terminálem připojíme na SQL server a na příkazový řádek bychom zadávali přímo příkazy jazyka SQL.

1.5.1 Rozdělení jazyka SQL

Jak už jsem se zmínil, SQL se skládá z několika částí. Některé části jsou určeny pro administrátory a návrháře databázových systémů, jiné pak pro koncové uživatele a programátory. První částí jazyka SQL je jazyk DDL - Data Definition Language. Jedná se o jazyk pro vytváření databázových schémat a katalogů. Způsob ukládání tabulek definuje jazyk SDL - Storage Definition Language. Třetí částí pro návrháře a správce je jazyk VDL - View Definition Language, určující vytváření pohledů (pohled si lze představit jako virtuální tabulku složenou z různých jiných tabulek). Poslední částí, kterou se budu převážně zabývat, je jazyk DML - Data Manipulation Language, který obsahuje základní příkazy INSERT, UPDATE, DELETE a nejpoužívanější příkaz SELECT. S jazykem DML pracují nejvíce koncoví uživatelé a programátoři databázových aplikací.

2. Problematika dopravy

Zde uvedu nejzákladnější veličiny popisující vlastnosti dopravního proudu.

2.1 Intenzita dopravního proudu

Jedna ze základních veličin popisující vlastnosti dopravního proudu je intenzita. Tato veličina nám poskytuje přehled o rozložení vozidel v čase. Jedná se o počet vozidel, která projedou měřeným bodem, za jednotku času.

$$q = \frac{n(t,x)}{t}$$

kde n(t,x) je počet vozidel projetých bodem x za čas t.
Pro stacionární proud je intenzita dopravního proudu převrácenou hodnotou průměrných časových intervalů mezi jednotlivými vozidly.

2.2 Hustota dopravního proudu

Další základní veličina popisující dopravní proud je hustota dopravního

proudu. Poskytuje informaci o počtu vozidel na jednotku délky měřeného úseku. Jedná se o počet vozidel která se v daném časovém okamžiku nacházejí v měřeném úseku.

$$\rho = \frac{n(t,x)}{x}$$

kde n(t,x) je počet vozidel v měřeném úsek v daném okamžiku t a x je délka úseku. Pro stacionární dopravní proud je hustota dopravního proudu odpovídá průměrným rozestupům mezi vozidly.

2.3 Obsazenost detektorů

Máme – li k dispozici intenzitu dopravního proudu a obsazenost detektorů, je možno určit dopravní stupeň. Obsazenost detektoru vyjadřuje poměr mezi dobou, kdy byl detektor obsazen vozidlem a dobou kdy nad detektorem nebylo žádné vozidlo. Může tedy nabývat 0 – 100 % doby měření. Máme – li tedy 1,5 minutová data, hodnota 50% vyjadřuje, že po dobu 45 sekund se nad detektorem nacházelo nějaké vozidlo. Právě dvojice Intenzita – Obsazenost se nejčastěji používá ke klasifikaci dopravy. Pokud bychom měli k dispozici jen intenzitu, nebylo by jasné, zda při Intenzitě = 0 je komunikace bez provozu, nebo zda je v místě měření stojící kolona. Algebraický vztah mezi Obsazeností τ a hustotou dopravního proudu ρ je následující:

$$\tau = (L + l)\rho$$

kde **L** je průměrná délka vozidel nad měřící smyčkou a **l** je délka měřící smyčky.

Podrobnější rozbor dopravní teorie lze nalézt v habilitační práci
Doc. Pavla Přibyla

3. Popis databázové části systému

Databázová část se skládá ze dvou částí. Jednak z databáze dopravních dat a dále pak z ovládacího programu.

3.1 Databáze dopravních dat

Databáze dopravních dat slouží k uchovávání historických dat, on-line dat, vyfiltrovaných dat, predikovaných dat a dále pak definic oblastí a seznamu křižovatek a detektorů. Původně bylo zamýšleno celý systém navrhnout pro on-line přísun dat pomocí FTP protokolu, leč jak se později ukázalo, poskytovatel dat není ještě na on-line zasílání dat připraven. On-line provoz je tedy simulován a on-line data jsou skladována v tabulce databáze k tomu určené.

Jako databáze může posloužit jakákoliv soudobá relační databáze. Zde je potřeba zmínit některé vlastnosti, které by v budoucnu mohly znepříjemnit ne-li znemožnit používání tohoto informačního systému jako celku. Měla by tedy splňovat následující kritéria:

- velikost zpracovatelného datového souboru
- nativní podpora správy uživatelů
- rychlost zpracování dat
- triggery
- storage procedury

3.1.1 Velikost zpracovatelného datového souboru

Velikost zpracovatelného datového souboru udává velikost datového souboru, který je daná databáze zpracovat. Tento parametr zde uvádím proto, že v této naší konkrétní aplikaci je klíčový. Pro ilustraci uvedu příklad. Mějme jednu křižovatku, ne které je umístěno devět čidel. Každé čidlo za 1,5 minuty poskytne hodnoty o intenzitě a obsazenosti.

Jedno čidlo představuje za jeden den $\frac{60}{1,5} * 24 = 960$ hodnot intenzity a 960 hodnot obsazenosti. Křižovatka o 9 čidlech pak za jeden den představuje $2 * 960 * 9 = 17280$ hodnot

Za jeden rok by to bylo 6307200 hodnot jen pro jednu křižovatku. Tomu odpovídá 496 MB veliký datový soubor. Ovšem jen za předpokladu, že se v něm nic jiného nenachází.

3.1.2 Nativní podpora správy uživatelů

Nativní podporou správy uživatelů rozumíme systém dohledu, omezování, povolování a správy uživatelů z hlediska jejich přístupů k jednotlivým objektům databázového systému. V tomto případě se jedná pouze o přístupy k tabulkám. V praxi to znamená, že uživatel s administrátorskými právy má přístup všude, může upravovat obsah i strukturu tabulek i celé databáze (tvorba a mazání celých tabulek), na rozdíl od běžného uživatele který má prává pouze Read-only. Ten bude moci tabulky pouze prohlížet. Pojem nativní podpora znamená, že systém má tuto podporu již vestavenu a není již ji potřeba doprogramovávat. V tomto našem systému by bylo možno zavést tři uživatele. A sice

- správce dat
- klasifikátor
- prediktor

kde správce dat by ukládal nová data, filtroval nová data a ukládal je do nové tabulky a mazal již neaktuální data. Klasifikátor by měl přístup jen do vyfiltrovaných dat a do oklasifikovaných dat a prediktor by měl přístup do oklasifikovaných dat a do predikovaných dat.

3.1.3 Rychlost zpracování dat

Rychlost zpracování dat bude velmi závislá na konkrétním řešení. Nejvíce ji bude ovlivňovat to, zda ovládací program a databáze budou na jednom počítači, nebo zda půjde o spojení přes síť. Vzhledem k dnešním rychlostem diskového rozhraní počítače a disků samotných bude toto “1-počítačové “ řešení rychlejší než

síťové řešení. Rychlost diskového rozhraní je dána komunikačním protokolem. Současné disky komunikují protokolem ATA-100 nebo ATA-133, jejichž maximální rychlost je 100 respektive 133 MB / s. Rychlost ethernetové sítě je 10 nebo 100 Mb / s. Vzhledem ke komunikačním protokolům ethernetové sítě (např. TCP/IP nebo IPX/SPX) se reálná rychlost pohybuje okolo 8 – 10 MB pro 100 Mb síť a 0,8 – 1 MB pro 10 Mb síť. Dále pak rychlost zpracování dat samozřejmě ovlivní i výkon samotného databázového serveru (software) i serveru (počítač) jako takového. Při síťovém řešení je ale nejužším hrdlem bezesporu síť samotná, takže výkon serveru (počítače) zas tak rozhodující nebude. Samozřejmě za předpokladu, že se nejedná o zastaralý a výkonově nevyhovující typ.

3.1.4 Triggery

Trigger je softwarový prostředek zajišťující elementární úkony na úrovní tabulky. Může jedna spouštět složité výpočtové procedury i jednoduché výpočty. Pro ilustraci uvedu příklad. Mějme sloupce tabulky Pole1, Pole2 a Pole3. Vyplním – li do Pole1 a Pole2 číselnou hodnotu, pak trigger v Poli2 může spustit výpočet:

Hodnota(Pole3) = Hodnota(Pole1) + Hodnota(Pole2)

Čímž ušetřím zadávání 3. hodnoty. Předpokladem je, že zadávaná hodnota Pole3 je algoritmizovatelná pro celou tabulku (ve všech záznamech bude Pole3 = Pole1 + Pole2).

3.1.5 Storage procedury

Storage procedury jsou programy uložené v databázi, které se dají spouštět externími programy. Mohou vykonávat funkce nad daty (kopírování, mazání, výpočty, aktualizace, atd.) nebo se mohou starat o údržbu celého databázového systému (zálohování, komprimace, přeindexování, obnovování, atd.). Storage procedury dále mohou poskytovat výsledky na základě dotazů do několika tabulek. Kromě jiného mohou být spouštěny pomocí triggerů, to znamená, že systém se dá za určitých podmínek plně zautomatizovat.

3.2 Vlastnosti databáze Microsoft Access

Velikost souboru databáze aplikace Microsoft Access (.mdb)	2 GB. Databáze však může obsahovat propojené tabulky v jiných souborech a její celková velikost je pak omezena jen kapacitou dostupné paměti.
Počet objektů v databázi	32 768
Počet otevřených tabulek	2048. Reálná hodnota může být nižší díky interně otevřeným tabulkám aplikace Microsoft Access.
Počet otevřených tabulek	2048. Reálná hodnota může být nižší díky interně otevřeným tabulkám aplikace Microsoft Access.
Počet polí v tabulce	255
Velikost tabulky	1 GB
Počet znaků v poli typu Poznámka	65 535 při vkládání dat prostřednictvím uživatelského rozhraní;1 GB při vkládání dat z programu.
Velikost pole typu Objekt OLE	1 GB
Počet současných uživatelů	255
Počet znaků v názvu uživatele nebo skupiny	20
Počet znaků hesla	14

3.3 SQL Server jako Alternativa k MS Access

3.3.1 Vysoký výkon a přizpůsobivost

V mnoha situacích aplikace Microsoft SQL Server nabízí lepší výkon než databáze Access. Aplikace SQL Server také podporuje velmi rozsáhlé databáze (až do **1 TB**). Nynější hranice velikosti databází Access je **2 GB**. Navíc aplikace SQL Server spolupracuje velmi účinně s operačním systémem Microsoft Windows NT. Dotazy provádí paralelně (k zpracování požadavků uživatelů využívá vícenásobné nativní podprocesy v rámci jednoho procesu) a pokud databázi používá více uživatelů, minimalizuje dodatečné požadavky na paměť.

3.3.2 Zvýšená dostupnost

Aplikace Microsoft SQL Server umožňuje v průběhu používání databáze dynamické zálohování (přírůstkové nebo úplné). Při zálohování dat tedy nemusíte nutit uživatele k ukončení práce s databází. To znamená, že databáze může být spuštěna 24 hodin denně, 7 dnů v týdnu.

3.3.3 Zlepšené zabezpečení

Aplikace Microsoft SQL Server může být propojena se zabezpečením operačního systému Windows NT a umožnit tak současné připojení k síti i k databázi. Správa komplexních zabezpečovacích schémat je proto jednodušší. Databáze SQL Server umístěná na serveru je lépe chráněna i z toho důvodu, že se neautorizovaní uživatelé nemohou připojit přímo k souboru databáze, ale musí se nejprve přihlásit k serveru.

3.3.4 Okamžitá obnovitelnost

Pro případ selhání systému (například při selhání operačního systému v důsledku výpadku napájení) má aplikace Microsoft SQL Server zabudován automatický mechanismus obnovy, který uvede databázi do posledního konzistentního stavu v průběhu několika minut, a to bez jakéhokoli zásahu správce systému. Důležité aplikace začnou opětovně fungovat téměř okamžitě.

3.3.5 Spolehlivé rozložení dat a transakcí

Zpracování transakcí je životně důležitým požadavkem systému, který má podporovat kritické aplikace, jako jsou například bankovní operace a zadávání objednávek v režimu on-line. Aplikace Microsoft SQL Server podporuje jednotlivé transakce pomocí protokolování transakcí. Tím je zajištěno, že veškeré změny provedené v průběhu transakce jsou buď potvrzeny nebo vráceny zpět.

Konzistence a obnovitelnost databázových transakcí je zaručena dokonce i v případě selhání systému a v průběhu úplné aktualizace prováděné více jak jedním uživatelem. Aplikace SQL Server provádí veškeré změny databáze v transakcích - základních jednotkách zpracování. Podle definice je buď bezpečně zakončena celá

transakce a výsledné změny se odrazí v databázi nebo je transakce vrácena a všechny změny v databázi jsou navráceny zpět.

Aplikace SQL Server používá dvoufázový protokol potvrzení. Může tak dokonce podporovat synchronizované transakce probíhající na více než jednom serveru a zajistit přitom, aby všechny servery v síti byly v konzistentním stavu.

3.3.6 Zpracování na serveru

Aplikace Microsoft SQL Server byla od začátku navržena jako databáze typu klient-server. Data s indexy jsou umístěna na jediném serveru, ke kterému má prostřednictvím sítě přístup mnoho klientských počítačů. SQL Server také snižuje zatížení sítě - databázové dotazy zpracuje přímo na serveru a teprve výsledky pošle klientovi. Aplikace typu klient-server zpracovává tedy úlohy tam, kde je to nejlepší - na serveru. Aplikace také může využívat uložené procedury a triggery. Pomocí nich se - namísto u klienta - na serveru centralizuje a sdílí logická struktura aplikace, obchodní pravidla a postupy, komplexní dotazy a kód pro ověřování dat a referenční integritu.

4. Realizace

4.1 Části systému

4.1.1 Ovládací program

Ovládací program zde plní funkci serveru, který zajišťuje zpracování on-line dat, jejich vyfiltrování a uložení do patřičných tabulek za účelem dalšího zpracování v externích modulech. V našem případě se jedná o jejich oklasifikování a případnou predikci na jejich základě. Ovládací program by mohl být včleněn do databáze jako jeho nedílná část, ale toto řešení, kdy je ovládací program oddělen od samotné databáze dat se mi jeví jako lepší řešení, neboť poskytuje v budoucnu vylepšit odděleně buď databází nebo program. Tím je do značné míry vyřešen problém s přenositelností dat na jinou platformu, neboť spojení Program <-> Databáze je uskutečněno pomocí DirectODBC rozhraní. To znamená, že data mohou být umístěna v lokální nebo vzdálené databázi a na situaci to nic nemění. Stejně tak nezáleží na tom, zda jsou data uložena v relativně jednoduché databázi Microsoft Access, která je určena spíše pro malé objemy dat, nebo v robusní databázi firmy Oracle. Na počítači, kde je umístěn ovládací program se jen vytvoří a patřičně nastaví Zdroj dat, což je systémový prostředek k vytvoření a specifikaci parametrů spojení s databází podporující různé komunikační protokoly. Výše zmíněné možnosti řešení umístění a typu databáze platí za předpokladu, že zůstanou zachovány názvy a struktura tabulek.

To samé pak platí i o ovládacím programu. Pokud se v budoucnu ukáže jeho současná forma jako nevyhovující, lze ho kompletně přepracovat, aniž by se musela jakkoliv měnit struktura či parametry databáze dopravních dat.

4.1.2 Klasifikační modul

Pomocí tohoto modulu dochází ke klasifikaci naměřených hodnot z jednotlivých čidel do formátu 5-ti stupňů popisujících dopravní zatížení daných úseků. Klasifikátor komunikuje s databází pomocí ODBC a po spuštění procedury čerpá potřebná data z vytvořeného dotazu. Získané hodnoty jsou pomocí

klasifikačního algoritmu převedena na stupně zatížení. Nově získané hodnoty (současný stav a predikce) jsou poslány zpět do databáze, kde jsou uchovány pro budoucí použití. Současně jsou poslány do výstupních modulů, které slouží pro konečné zobrazení dat (grafické i textové).
Pro realizaci tohoto modulu je použito programovacího prostředí Visual Basic 6.0.

4.1.3 Predikční modul

Tento modul využívá data z databáze nebo z modulu pro klasifikaci dat k predikci hodnot pro daný časový horizont. Tento časový horizont bude závislý hlavně na intervalu mezi jednotlivými měřeními přicházejícími do datového skladu. Zatím se předpokládá predikování dat nanejvýš na 30 minut. Predikce může být realizována buď na základě již oklasifikovaných dat z klasifikačního modulu a nebo z naměřených dat z datového skladu, kdy ke klasifikaci dojde až po průchodu dat prediktorem.
Při tvorbě tohoto modulu bude použito programovacího prostředí Visual Basic 6.0 spolu s programem NeuroSolution.

4.1.4 Moduly pro zobrazení dat

Tyto moduly mají za úkol poskytnout již dříve zmiňovaná oklasifikovaná data a ta v co nejsrozumitelnější formě podat koncovému uživateli. Při realizaci tohoto projektu se počítá s následujícími výstupními moduly:

4.1.4.1 WWW aplikace

Tento modul graficky zobrazuje dopravní zatížení na jednotlivých úsecích zpracovávaného území. Jednotlivé stupně zatížení jsou odlišeny barevně společně s číselným vyjádřením dané situace. Do mapy je bude možné promítnout predikované hodnoty, aby byl uživatel informován, jak se situace změní v horizontu např. 15 a 30 minut. Změna měřítka pro detailnější přiblížení je také jedna z funkcí tohoto modulu. Tento modul běží na webovém serveru Apache a pro programování je použito skriptovacího jazyku PHP. Tento software je volně šiřitelný což zaručuje

jak další vývoj tohoto programu tak v podstatě nulové náklady. Tento jazyk si navíc velice dobře rozumí právě s webovým serverem Apache.

4.1.4.2 Wap aplikace

Tento modul by měl zobrazovat stupně zatížení dopravy (získaná pravděpodobně ve formátu XML), která mu pošle klasifikační modul, na mobilních telefonech. Formát zobrazení dat bude adekvátní možnostem mobilního telefonu. Data tedy nebudou mít takovou grafickou podobu jako u Web aplikace, na druhou stranu budou dostupnější díku zobrazovacímu médiu a jeho velkému rozšíření. Poslední trendy však ukazují na jistou stagnaci Wap aplikací, protože mnoho nových typů mobilních telefonů interně podporuje přístup na internet pomocí WWW prohlížeče místo Wap rozhraní. To umožňují zejména nové mobilní technologie jako jsou GPRS nebo HSCSD.

4.1.4.3 Desktopová aplikace

Jde o modul nainstalovaný na počítači, který bude sloužit pro grafické zobrazení dat a získání podrobnějších informací. Tento modul bude mít v sobě funkce pro zobrazení historie dat s možností výpočtu potřebných statistických parametrů, porovnání skutečné a predikované hodnoty a podobně. Tento modul nebude pravděpodobně volně dostupný pro běžného uživatele. Bude využíván v rámci dopravní fakulty ČVUT a pro použití dopravní policie ČR.

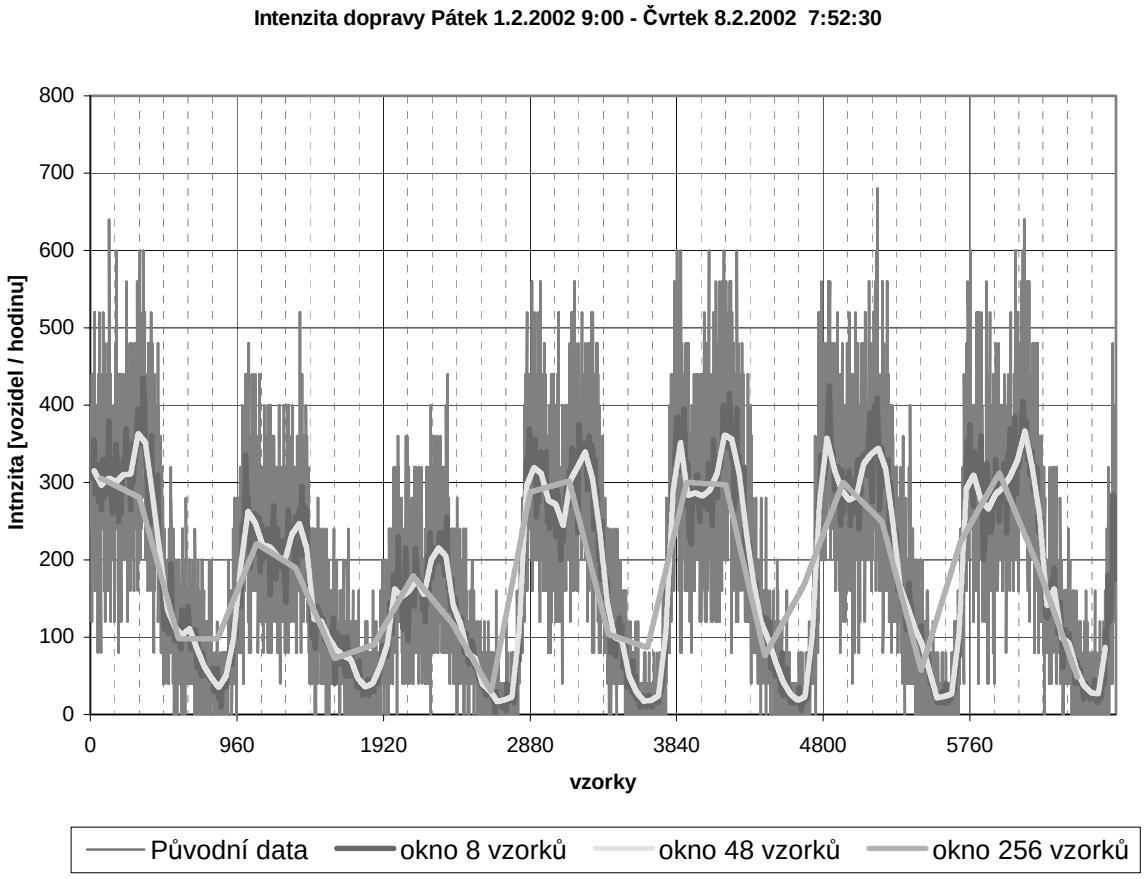
4.1.4.4 Portál

Tento modul by měl předávat potřebné informace prostřednictvím informačních tabulí nebo portálů. Informace budou tedy dostupná i pro řidiče jedoucího do dané oblasti i bez použití mobilního telefonu.

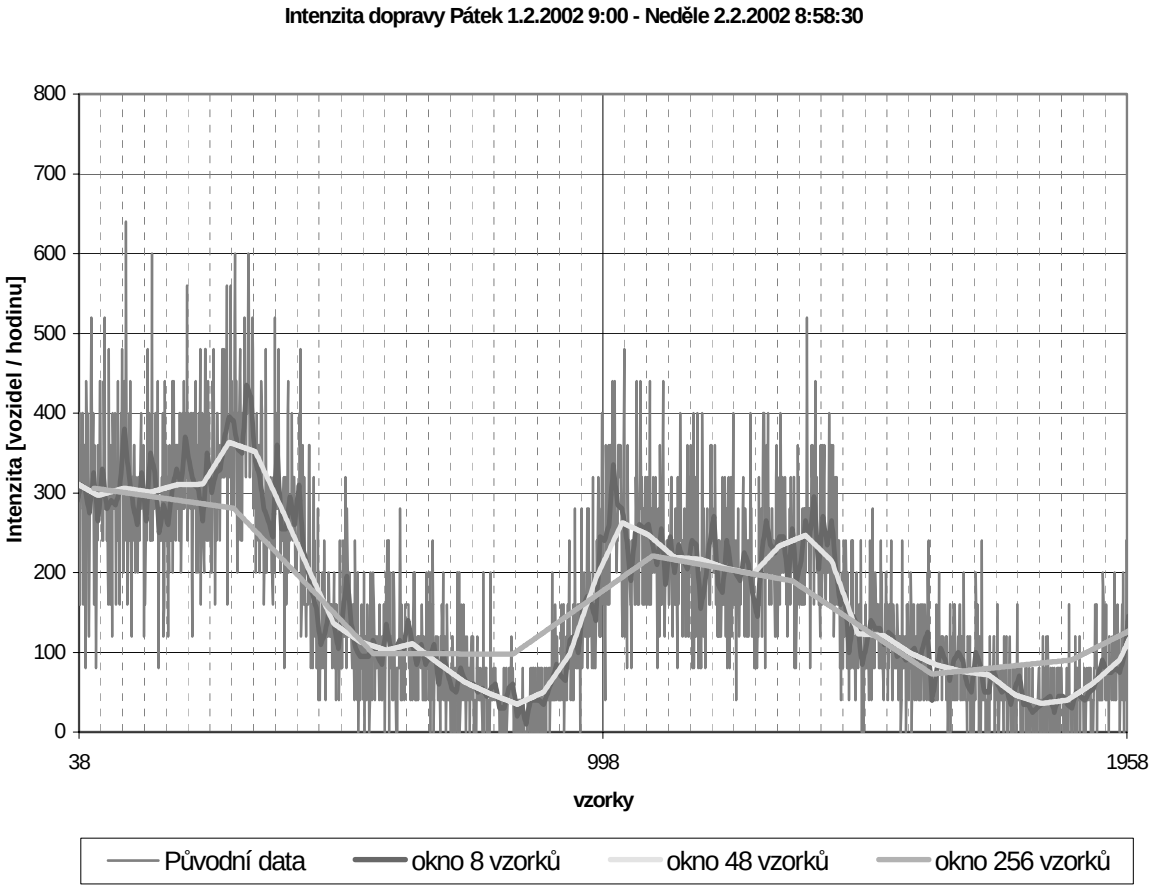
4.2 Filtrování dat

V kapitole Databáze dopravních dat jsem popsal, jak bude zhruba veliký datový soubor pro jednu křižovatku za rok. V této aplikaci uvažujeme ale křižovatek

hned několik a tudíž by zákonitě došlo k neúnosnému zpomalení celého systému. Vybírat určité konkrétní záznamy v několika miliónech jiných záznamů již nějakou dobu trvá. Proto jsem se, po konzultaci s ostatními kolegy, kteří pracují na predikčním a klasifikačním modulu, rozhodl, že surová dopravní data zbavím šumu a proložím je vhodnou křivkou, jejíž popis budu skladovat. Jako nejvhodnější metoda pro vyhlazení zašuměných dopravních dat se mi jevila metoda klouzavých průměrů. Nakonec se ale ukázalo, že postačí pouze obyčejný průměr za 8 po sobě jdoucích vzorků. Hodnota tohoto průměru je pak skladována v tabulce a dále pak poskytována k dalšímu zpracování. Pokud by byly potřeba hodnoty mezi těmito průměry, lze je pak snadno dopočítat lineární regresí. Takto vzniklá lomená čára dobře popisuje jak trendy, tak neočekávané výkyvy v hodnotách intenzity a obsazenosti. Požijeme-li větší šířku okna, ze kterého se průměr počítá, dostaneme lomenou čáru lépe popisující trendy, ale odezva systému by se prodloužila na neúnosnou mez.



graf 1: Intenzita dopravy pro data 6. týdne.



graf 2: Intenzita dopravy pro data 6. týdne – detail

Obdobně se chová i vyhlazování obsazenosti. Z grafů je patrné, že nejlepší volbou by zřejmě bylo zřejmě okno o šířce 48 vzorků. Tato lomená čára nejlépe vystihuje trendy a i lokální výkyvy popisuje docela dobře. Pokud ale nastane neočekávaná nestandardní situace, jako například nehoda na křižovatce, příjezdové vozovky se skokově zaplní. Systém by ale při šířce okna 48 vzorků na to zareagoval až za

48 * 1,5 min. = 72 minut

což je pro kvalitní řízení dané oblasti i pro včasné informování řidičů naprosto nedostačující.

4.3 Seznam tabulek databáze

Dopravni_data_surova
Dopravni_data_surova_online
Dopravni_data_filtrovana
Dopravni_data_predikovana
Dopravni_stupne
seznam_idK_idD
definice_oblasti
stavy_systemu

4.3.1 Tabulka Dopravni_data_surova

Struktura tabulky Dopravni_data_surova:

název pole	datový typ	Formát	rozsah
Datum	datum/čas	datum (krátké)	1.1.100 až 31.12.9999
Cas	datum/čas	datum (krátké)	0:00:00 až 23:59:59
Id_krizovatky	Text		255 znaků
Id_detektoru	Text		255 znaků
Intenzita	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)
Obsazenost	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)

Ukázka dat tabulky Dopravni_data_surova:

datum	cas	id_krizovatky	id_detektoru	intenzita	obsazenost
21.12.2001	8:00:00	M14	DVA(468)	520	66
21.12.2001	8:01:30	M14	DVA(468)	200	66
21.12.2001	8:03:00	M14	DVA(468)	240	66
21.12.2001	8:04:30	M14	DVA(468)	240	66
21.12.2001	8:06:00	M14	DVA(468)	200	66
21.12.2001	8:07:30	M14	DVA(468)	200	66
21.12.2001	8:09:00	M14	DVA(468)	320	22
21.12.2001	8:10:30	M14	DVA(468)	440	66
21.12.2001	8:12:00	M14	DVA(468)	240	66
21.12.2001	8:13:30	M14	DVA(468)	240	66
21.12.2001	8:15:00	M14	DVA(468)	280	55

Tabulka Dopravni_data_surova je do databáze začleněna jako datový sklad

naměřených dopravních dat, která v budoucnu budou přicházet on-line. Ovládací program z ní načítá vždy určitý počet časově po sobě jdoucích dat, která se přesunují do tabulky Dopravni_data_surova_online, kde se s nimi dále pracuje. V budoucnu se tato tabulka nebude používat, protože on-line data se budou načítat rovnou do tabulky Dopravni_data_surova_online.

4.3.2 Tabulka Dopravni_data_surova_online

Struktura tabulky Dopravni_data_surova:

název pole	datový typ	formát	rozsah
Datum	datum/čas	datum (krátké)	1.1.100 až 31.12.9999
Cas	datum/čas	datum (krátké)	0:00:00 až 23:59:59
id_krizovatky	Text		255 znaků
id_detektoru	Text		255 znaků
Intenzita	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)
Obsazenost	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)

Ukázka dat z tabulky Dopravni_data_surova_online:

datum	cas	id_krizovatky	id_detektoru	intenzita	obsazenost
21.12.2001	8:00:00	M14	DVA(468)	520	66
21.12.2001	8:01:30	M14	DVA(468)	200	66
21.12.2001	8:03:00	M14	DVA(468)	240	66
21.12.2001	8:04:30	M14	DVA(468)	240	66
21.12.2001	8:06:00	M14	DVA(468)	200	66
21.12.2001	8:07:30	M14	DVA(468)	200	66
21.12.2001	8:09:00	M14	DVA(468)	320	22
21.12.2001	8:10:30	M14	DVA(468)	440	66

Tabulka Dopravni_Data_surova_online slouží k dočasnému uchovávání surových dopravních dat, která byla systému zaslána. Když jich je určitý počet (specifikováno v tabulce stavy_systemu), ovládací program je vyfiltruje a uloží do tabulky Dopravni_data_filtrovana. Tato tabulka se po přesunu dat do tabulky Dopravni_data_vyfiltrovana automaticky zcela smaže, aby datový soubor neobsahoval zbytečné informace.

4.3.3 Tabulka Dopravni_data_filtrovana

Struktura tabulky Dopravni_data_filtrovana:

název pole	datový typ	Formát	rozsah
Datum	datum/čas	datum (krátké)	1.1.100 až 31.12.9999
Cas	datum/čas	datum (krátké)	0:00:00 až 23:59:59
Id_krizovatky	Text		255 znaků
Id_detektoru	Text		255 znaků
Intenzita	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)
Obsazenost	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)

Ukazka dat z tabulky Dopravni_data_filtrovana:

datum	cas	id_krizovatky	id_detektoru	intenzita	obsazenost
21.12.2001	8:06:00	M14	DVA(468)	295	60

Do této tabulky jsou uložena vyfiltrovaná data, která zde budou čekat na převzetí klasifikačním a predikčním modulem. Pokud nebudeme brát v úvahu tabulku Dopravni_data_surova, která bude v budoucnu z databáze vypuštěna, je tabulka Dopravni_data_filtrovana centrálním datovým skladem a v datovém souboru bude zabírat nejvíce místa. Všechny ostatní tabulky budou mít oproti této tabulce zanedbatelnou velikost. Pro účely predikce, respektive predikčního algoritmu, zde bude skladována dostatečně dlouhá časová posloupnost dat. Čím delší, tím lépe bude prediktor naučen a tím lepší bude poskytovat predikci.

4.3.4 Tabulka Dopravni_data_predikovaná

Struktura tabulky Dopravni_data_predikovana:

název pole	datový typ	Formát	rozsah
Datum	datum/čas	datum (krátké)	1.1.100 až 31.12.9999
Cas	datum/čas	datum (krátké)	0:00:00 až 23:59:59
Id_krizovatky	Text		255 znaků
Id_detektoru	Text		255 znaků
Intenzita	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)
Obsazenost	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)

Ukazka dat z tabulky Dopravni_data_predikovana:

datum	cas	id_krizovatky	id_detektoru	intenzita	obsazenost
8.2.2002	8:00:00	M14	DVA(468)	325	62
8.2.2002	8:01:30	M14	DVA(468)	307	62
8.2.2002	8:03:00	M14	DVA(468)	345	61
8.2.2002	8:04:30	M14	DVA(468)	316	61
8.2.2002	8:06:00	M14	DVA(468)	330	61
8.2.2002	8:07:30	M14	DVA(468)	342	62
8.2.2002	8:09:00	M14	DVA(468)	379	62
8.2.2002	8:10:30	M14	DVA(468)	340	61
8.2.2002	8:12:00	M14	DVA(468)	363	60
8.2.2002	8:13:30	M14	DVA(468)	332	60
8.2.2002	8:15:00	M14	DVA(468)	310	62

Zde se budou ukládat výsledky činnosti prediktoru. Časová řada Vyfiltrovaná data – Predikovaná data bude samozžejmě navazovat. Predikovaná data budou ovšem také sloužit k určení dopravního stupně v aktuálním čase. Naměřená data už jsou samozřejmě v tu dobu k dispozici, ale nejsou ještě vyfiltrována. Proto se bude vycházet z predikovaných dat a lineární regresí se současnost dopočítá. Zároveň je tak možno na jeden “filtrační“ krok zpět určovat přesnost predikce a případně z ní vyvodit důsledky.

4.3.5 Tabulka Dopravni_stupne

Struktura tabulky Dopravni_stupne:

název pole	datový typ	formát	rozsah

Datum	datum/čas	datum (krátké)	1.1.100 až 31.12.9999
Cas	datum/čas	datum (krátké)	0:00:00 až 23:59:59
Id_krizovatky	Text		255 znaků
Id_detektoru	Text		255 znaků
dopravni_stupen_klasifikovany	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)
Dopravni_stupen_predikovany	Číslo	dlouhé celé číslo	4 byty (-2,147,483,648 až 2,147,483,647)

Ukázka dat z tabulky Dopravni_stupne:

Datum	cas	id_krizovatky	id_detektoru	dopr_stupen_klas	dopr_stupen_pred
21.12.2001	8:00:00	M14	DVA(468)	5	3
21.12.2001	8:01:30	M14	DVA(468)	2	2
21.12.2001	8:03:00	M14	DVA(468)	2	2
21.12.2001	8:04:30	M14	DVA(468)	2	2
21.12.2001	8:06:00	M14	DVA(468)	2	2
21.12.2001	8:07:30	M14	DVA(468)	2	2
21.12.2001	8:09:00	M14	DVA(468)	3	3
21.12.2001	8:10:30	M14	DVA(468)	4	3
21.12.2001	8:12:00	M14	DVA(468)	2	3
21.12.2001	8:13:30	M14	DVA(468)	2	2
21.12.2001	8:15:00	M14	DVA(468)	3	3

Zde se budou nacházet výsledky z klasifikačního modulu. Ten bude brát data z tabulky Dopravni_data_filtrovana pro minulost a z tabulky Dopravni_data_predikovana pro současnost nebo budoucnost. Tato tabulka bude obsahovat pouze ta data, která slouží jako podklad k uspokojení požadavku, který vstoupil do systému prostřednictvím některé z on-line aplikací prezentující výsledky koncovým uživatelům.

4.3.6 Tabulka seznam_idK_idD

Struktura tabulky seznam_idK_idD:

název pole	datový typ	formát	rozsah
id_krizovatky	Text		255 znaků

id_detektoru	Text		255 znaků

Ukázka dat z tabulky seznam_idK_idD:

id_krizovatky	id_detektoru
M14	DVA(468)
M14	DVA1(466)
M14	DVA1a(467)
M14	DVAa(469)
M14	DVB(470)
M14	DVBa(471)
M14	DVBb(472)
M14	S15(473)
M14	S16(474)

Toto je jedna z klíčových tabulek systému. Zde se nachází seznam zaregistrovaných detektorů, potažmo křižovatek, které bude systém akceptovat. Vyskytne-li se v přišidších datech id křožovatky nebo id detektoru, které systém nezná, nebudou tato data zpracovávána. Tím, že jako id slouží řetězce a ne jen čísla, je na první pohled zřejmé o jaká data se jedná, nicméně za to platíme určitým snížením výkonu celého systému.

4.3.7 Tabulka Definice_oblasti

Struktura tabulky Definice_oblasti:

název pole	datový typ	Formát	rozsah
nazev	text		255 znaků

id_krizovatky	Text		255 znaků
id_detektoru	Text		255 znaků
koeficient	číslo	jednoduchá přesnost	–3.402823+e38 až –1.401298+e–45 pro záporná čísla a 1.401298+e–45 až 3.402823+e38 pro kladná čísla.

Ukázka dat z tabulky Definice_oblasti

nazev	id_krizovatky	id_detektoru	koeficient
Ječná	M14	DVA1(466)	0,4
Ječná	M15	DVAa(486)	0,35
Ječná	M510	DVAc(175)	0,2
Ječná	M573	DVE1a(560)	0,05

Toto je další klíčová tabulka. Společně s předchozí dává kompletní publikovatelný výstup, neboť kombinací této a předchozí vznikne údaj **Ječná -> 5**, který
Ing. Stareček požívá ve své práci zabývající se prezentací těchto dat na internetu.

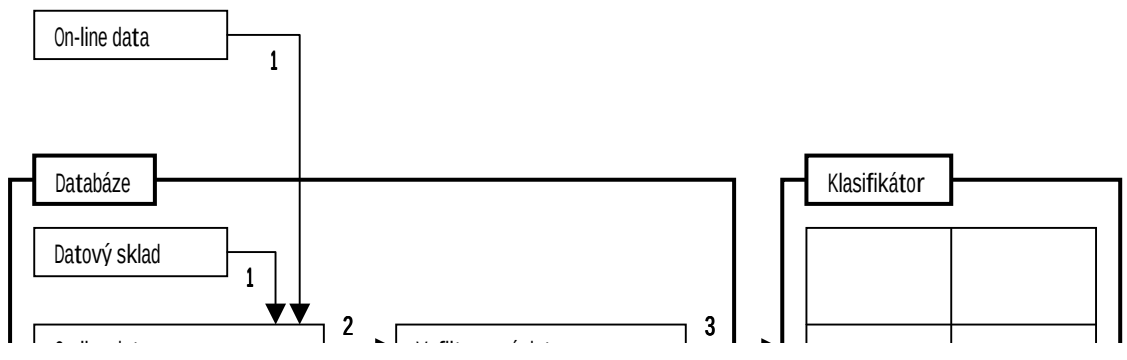
4.3.8 Tabulka stavy_systemu

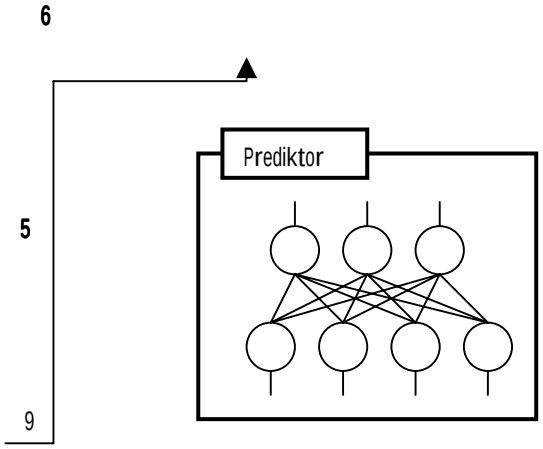
Tabulka_stavy_systemu obsahuje pole sloužící jako globální proměnné společné pro všechny části systému. Jedná hlavně se o pole:

- pocet_vzorku_na_okno
- delka vzorku
- nova_data_počet

Tato pole určují parametry filtrování. Další pole mohou určovat parametry klasifikace a predikce. Tato tabulka se však stále ještě mění a její struktura se i v budoucnu bude přizpůsobovat nárokům, které na ni budou klást jednotlivé moduly systému

4.4 Schéma datových toků





4.4.1 Popis datových toků

(1) - Příchozí on-line nebo skladovaná on-line data. Formát bude pravděpodobně plain text nebo tabulka programu Excel

(2) - Proces filtrování dat. Zde se dá hovořit o kompresi dat v poměru 1 : Stavy_systemu(počet_vzorku_na_okno) v našem případě máme nastaveno 8.

(3) - Vyfiltrovaná data přebírá klasifikační modul. Ke klasifikaci je možno použít standardní shlukové analýzy nebo méně obvyklých metod jako jsou neuronové sítě.

(4) - Vyfiltrovaná data přebírá predikční modul. K predikci se využívá vrstevnatá neuronová síť se systémem učení typu Back-propagation.

(5) - Prediktor vrací predikovaná data do databáze, aby mohla být oklasifikována klasifikátorem.

(6) - Klasifikátor přebírá predikovaná data. Predikovaná data mají stejnou formu jako data vyfiltrovaná, tudíž není nutné mít jiný typ klasifikátiru.

(7) - Na základě externího požadavku **(9)** jsou data z tabulek Dopravni_data_vyfiltrovana a Dopravni_data_predikovana předána do tabulky Dopravni_stupne.

(8) - Datový tok **(7)** je zkombinován s informacemi o oblastech z tabulky Definice_oblasti a ve vhodné formě předána klientovi. (www prohlížeč, wap prohlížeč, portál). Formát je plain text nebo XML.

(9) - Dotaz on-line klienta na data.

Formát datových toků **(2)** – **(8)** je buď sql dotaz nebo sql tabulka.

4.5 Okomentovaný zdrojový kód

Zde uvádím okomentovaný zdrojový kód hlavních subrutin zajišťující jednak import off-line dat z tabulek programu Excel a dále pak dvě hlavní subrutiny zajišťující filtrování dat.

4.5.1 Import off-line dat

Problém spočívá v tom, že každý .XLS soubor obsahuje v řádku 1 popis sloupců. Ten je ve tvaru: **M14.DVA(468) / Zählung (Fz/h)** Po načtení ovládacím programem se však tečka přetransformuje na dvojitý křížek, protože tečka v názvu pole je nepovolený znak. Řetězec **M14#DVA(468) / Zählung (Fz/h)** tedy obsahuje dva údaje, které identifikují data v tomto sloupci tabulky. Ty je potřeba z něj získat. Jedná se o **M14** a **DVA(468)**. Problém je, že ani jeden z nich nemá pevnou délku.

```
Private Sub Command1_Click()

zacatek = Timer

Me.data1 je reprezentace XLS tabulky s hodnotami intenzit
```

Me.data2 je reprezentace XLS tabulky s hodnotami obsazenosti

```
Me.Data1.Refresh
Me.Data2.Refresh

Me.Data1.Recordset.MoveFirst
Me.Data2.Recordset.MoveFirst
```

zjistím id_krizovatky a jeho délku. To provedu pomocí cyklu, který kontroluje řetězec znak po znaku zleva a hledá znak "#"

```
nazev = Me.Data1.Recordset.Fields(2).Name
For i = 2 To Len(nazev)
  If Mid(nazev, i, 1) = "#" Then id_krizovatky = Left(nazev, i - 1)
Next i
```

Zde plním tabulku Dopravni_data_surova údaji z načtených .XLS souborů. Jeden pro intenzitu a jedne pro obsazenost.

```
Do Until Me.Data1.Recordset.EOF

  For sloupec = 1 To Me.Data1.Recordset.Fields.Count - 1

    id_detektoru = Mid(Me.Data1.Recordset.Fields(sloupec).Name,
    Len(id_krizovatky) + 2, Len(Me.Data1.Recordset.Fields(sloupec).Name) –
    Len(id_krizovatky) - 18)
```

Do proměnné id_detektoru dám hodnotu z názvu sloupce ohraničenou zleva délkou id_krizovatky + 2. Od tohoto bodu pak doprava celkovou délkou řetězce zkrácenou o délku id_krizovatky a o 18 znaků. To proto, protože na konci řetězce je řetězec ***/ Zählung (Fz/h)*** *konstantní délky 18 znaků.*
Me.rst_surova_data je reprezentace tabulky Dopravni_data_surova.

```
    Me.rst_surova_data.Recordset.AddNew
```

```
        Me.rst_surova_data.Recordset!cas =
        TimeValue(Me.Data1.Recordset.Fields("F1").Value)

        Me.rst_surova_data.Recordset!datum =
        DateValue(Me.Data1.Recordset.Fields("F1").Value)

        Me.rst_surova_data.Recordset!id_krizovatky = id_krizovatky

        Me.rst_surova_data.Recordset!id_detektoru = id_detektoru

        Me.rst_surova_data.Recordset!intenzita =
        Me.Data1.Recordset.Fields(sloupec).Value

        Me.rst_surova_data.Recordset!obsazenost =
        Me.Data2.Recordset.Fields(sloupec).Value

        Me.rst_surova_data.Recordset.Update
```

Zde se bere další sloupec XLS tabulky.

```
    Next sloupec
```

V obou XLS tabulkách se posunu na další řádek.

```
    Me.Data1.Recordset.MoveNext
    Me.Data2.Recordset.MoveNext

Loop

MsgBox "Úloha zpracována za " & Timer - zacatek & " s.", vbOKOnly, ""

End Sub
```

4.5.2 Filtrování dat – příprava

```
Sub kopiruj()

Zde se maže tabulka Dopravni_data_surova_online pomocí sql přkazu. Aby bylo
možno využít služeb sql interpretru, je nutné s tabulkami pracovat přes proměnnou
typu workspace a database.

Set wrkODBC = CreateWorkspace("", "", "", dbUseODBC)
Set databaze = wrkODBC.OpenDatabase("Dopravni_data_ACCESS",
dbDriverNoPrompt)

databaze.Execute "delete * from dopravni_data_surova_ONLINE"

databaze.Close
wrkODBC.Close

Me.rst_online_data.Refresh
Me.DataGrid2.ReBind

Me.rst_surova_data.Recordset.MoveFirst
Me.rst_surova_data.Recordset.Move Me.Text3.Text - 1

Podle počtu záznamů v tabulce Seznam_idK_a_idD určím počet detektorů. Dále pak
z tabulky stavy_systemu načtu  počet_vzorku_na_okno. Vynásobením těchto dvou
hodnot dostanu počet záznamů, které se mají zkopírovat.

Me.rst_seznam_idK_a_idD.Recordset.MoveLast
Me.rst_Stavy_systemu.Recordset.MoveFirst
For i = 1 To
(Me.rst_seznam_idK_a_idD.Recordset.RecordCount *
Me.rst_Stavy_systemu.Recordset!pocet_vzorku_na_okno)
Step 1
```

```
    Me.rst_online_data.Recordset.AddNew

    Me.rst_online_data.Recordset!datum = Me.rst_surova_data.Recordset!datum

    Me.rst_online_data.Recordset!cas = Me.rst_surova_data.Recordset!cas

    Me.rst_online_data.Recordset!id_krizovatky =
    Me.rst_surova_data.Recordset!id_krizovatky

    Me.rst_online_data.Recordset!id_detektoru =
    Me.rst_surova_data.Recordset!id_detektoru

    Me.rst_online_data.Recordset!intenzita =
    Me.rst_surova_data.Recordset!intenzita

    Me.rst_online_data.Recordset!obsazenost =
    Me.rst_surova_data.Recordset!obsazenost

    Me.rst_online_data.Recordset.Update

    Me.rst_surova_data.Recordset.Move 1

Next i

Me.Text3.Text = Me.Text3.Text + i - 1

End Sub
```

4.5.3 Filtrování dat – výpočet

```
Sub filtruj_data()

Dim cas_stred As Date
```

Me.rst_Stavy_systemu.Recordset.MoveFirst
Me.rst_online_data.Recordset.MoveFirst

Datum pro sql dotazy nemá běžně používaný tvar 31.12.2001 ale #31/12/2001#. Ten se ale u různých interpretrů sql jazyka může lišit. Běžně používaný je i tvar #12/31/2001#.

datum = "**#**" & **Day**(Me.rst_online_data.Recordset!datum) & "**/**" &
Month(Me.rst_online_data.Recordset!datum) & "**/**" &
Year(Me.rst_online_data.Recordset!datum) & "**#**"

pocet_vzorku = Me.rst_Stavy_systemu.Recordset!pocet_vzorku_na_okno
delka_vzorku = Me.rst_Stavy_systemu.Recordset!delka_vzorku

Určení časového středu okna.

cas_stred = Format("0:0" & (pocet_vzorku / 2) * delka_vzorku & ":00", "hh:mm:ss")
cas_stred = Me.rst_online_data.Recordset!cas + cas_stred

Set wrkODBC = CreateWorkspace("", "", "", dbUseODBC)
Set databaze = wrkODBC.OpenDatabase("Dopravni_data_ACCESS",
dbDriverNoPrompt)

Me.rst_seznam_idK_a_idD.Recordset.MoveFirst
Do Until Me.rst_seznam_idK_a_idD.Recordset.EOF

Proměnná sql_kod bude po určení hodnot všech dílčích proměnných obsahovat pole id_krizovatky, id_detektoru, konstantní pole cas_stred, vypočítávané pole prumer_I a vypočítávané pole prumer_O. Vycházet se bude z tabulky Dopravni_data_surova_ONLINE a budou se brát pouze řádky mající v polích id_krizovatky, id_detektoru a datum správné hodnoty. Výsledný výběr je sdružován

podle polí id_krizovatky, id_detektoru a datum.

sql_kod = "SELECT id_krizovatky, id_detektoru, "' & cas_stred & "' as cas, datum, ((Sum(intenzita))/" & pocet_vzorku & ") AS prumer_I, ((Sum(obsazenost))/" & Pocet_vzorku & ") AS prumer_O FROM Dopravni_data_surova_ONLINE WHERE (id_krizovatky="' & Me.rst_seznam_idK_a_idD.Recordset!id_krizovatky & "') AND (id_detektoru="' & Me.rst_seznam_idK_a_idD.Recordset!id_detektoru & "') AND (datum=" & datum & ") GROUP BY id_krizovatky, id_detektoru, datum"

Při otevření dotazu rs_tmp_prumery se jako zdroj záznamů uvádí výše uvedená proměnná sql_kod. Tím je ovládací program prostřednictvím sql příkazu donucen vypočítat řádky již při otevření a odpadá tím nutnost dalšího algoritmu pro výpočet sum na základě podmínek. Tento sql kód pracuje s parametry a je tudíž univerzálně použitelný.

Set **rs_tmp_prumery** = databaze.OpenRecordset(**sql_kod**, dbOpenDynaset)

Me.rst_vyfiltrovana_data.Recordset.AddNew

Me.rst_vyfiltrovana_data.Recordset!**datum** = rs_tmp_prumery!datum
Me.rst_vyfiltrovana_data.Recordset!**cas** = rs_tmp_prumery!cas
Me.rst_vyfiltrovana_data.Recordset!**id_krizovatky** = rs_tmp_prumery!id_krizovatky
Me.rst_vyfiltrovana_data.Recordset!**id_detektoru** = rs_tmp_prumery!id_detektoru
Me.rst_vyfiltrovana_data.Recordset!**intenzita** = rs_tmp_prumery!prumer_I
Me.rst_vyfiltrovana_data.Recordset!**obsazenost** = rs_tmp_prumery!prumer_O

Me.rst_vyfiltrovana_data.Recordset.Update

rs_tmp_prumery.Close

Následně se pak sql příkazem maže tabulka Dopravni_data_surova_online. Ovšem pouze řádky splňující zadané podmínky. Prakticky se ale mažou řádky všechny, neboť jiné záznamy než ty které splňují podmínky se v této tabulce stejně nenacházejí.

```
databaze.Execute "delete * from dopravni_data_surova_ONLINE WHERE
(id_krizovatky='" & Me.rst_seznam_idK_a_idD.Recordset!id_krizovatky & "') AND
(id_detektoru='" & Me.rst_seznam_idK_a_idD.Recordset!id_detektoru & "')"
```

Zde následuje úsek kódu zajišťující obnovu objektů na formuláři.

```
Me.rst_online_data.Refresh
Me.rst_vyfiltrovana_data.Refresh

Me.DataGrid2.ReBind
Me.DataGrid3.ReBind

Me.rst_seznam_idK_a_idD.Recordset.MoveNext
Loop
```

Opět úsek kódu zajišťující obnovu objektů na formuláři.

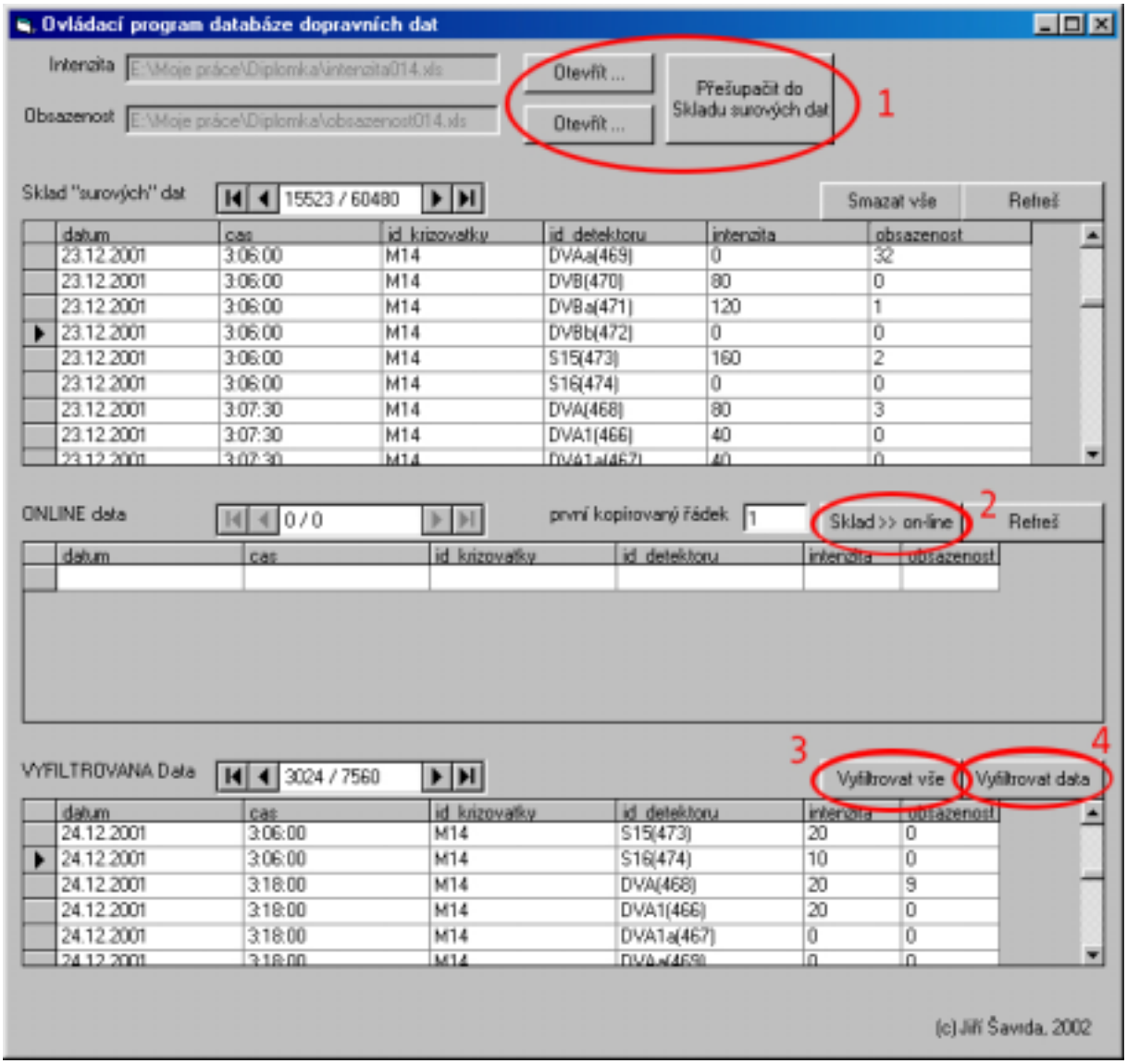
```
Me.rst_online_data.Refresh
Me.rst_vyfiltrovana_data.Refresh
Me.DataGrid2.ReBind
Me.DataGrid3.ReBind
Me.rst_Stavy_systemu.Refresh

databaze.Close
wrkODBC.Close

Me.rst_vyfiltrovana_data.Recordset.MoveLast

End Sub
```

4.6 Formulář ovládacího programu



4.6.1 Popis ovládacích prvků

(1) - Obě tlačítka **Otevřít …** otevřou standardní dialogové okno systému windows, pomocí kterého lze procházet disky i síť. V tomto případě se zadává soubor .XLS s hodnotami intenzit nebo obsazeností

(2) - Tlačítko **Sklad >> on-line** přesouvá v našem případě 72 záznamů z tabulky Dopravni_data_surova do tabulky Dopravni_data_surova_online. Začíná na pozici, která je specifikována ve vedlejším textovém poli. Číslo 72 dostaneme, vynásobíme-li počet detektorů zaregistrovaných v systému (tabulka seznam_idD_a_idK) počtem vzorků na okno (tabulka stavy_systemu).

(3) - Tímto tlačítkem spustím filtraci všech dat, která se nacházejí v tabulce Dopravni_data_surova. Z této tabulky se data začnou v blocích po 72 záznamech přesouvat do tabulky Dopravni_data_surova_online (simulace on-line zasílání dat). Poté se vyfiltrují a uloží do tabulky dopravni_data_filtrovaná. Tabulka Dopravni_data_surova_online se smaže a celý proces se opakuje, dokud nejsou všechna data z tabulky Dopravni_data_surova přenesena.

(4) - Toto tlačítko je zjednodušená verze předchozího. Předpokládá, že jsme pomocí **(2)** nějaká data již přesunuli a spustí filtrování těchto dat, která jsou po skončení tohoto procesu uložena.

5. Závěr

5.1 Výhody

Jako výhodu celého systému vidím jednoznačně v jeho modularitě. Ukáže-li se v budoucnu jakákoliv jeho část jako nevyhovující, není žádný problém ji vyměnit, bez nutnosti zásahu do systému jako celku. Změní-li se například forma dodávaných on-line dat ze současného .XLS formátu na jiný, doplní se patřičné funkce do ovládacího programu databáze, přičemž tyto změny nijak neovlivní ostatní moduly. Další výhodou je bezesporu nenáročnost na hardware počítače na kterých je tento systém provozován. Databázi Microsoft Access 2000 lze provozovat na běžném kancelářském PC. Stejně tak i ostatní moduly nevyžadují pro svůj chod počítače nejsilnější kategorie. Další výhodou systému jako celku je to, že výsledky jsou dostupné bez jakéhokoliv speciálního prohlížecího programu. Odpadá tudíž nutnost, aby si koncový uživatel musel na svůj počítač instalovat dodatečný software. Tím jsou výsledky dostupné široké veřejnosti a jakýkoliv počítač, který je vybaven připojením k internetu a patřičným internetovým prohlížečem (standadní součást všech systému Windows), je schopen je zobrazit.

5.2 Nevýhody

Nevýhodou systému by mohlo být to, že databáze Access 2000 má omezenu velikost datového souboru na 2 GB. Výhledově by tedy mohlo být nutné přejít na jinou databázovou platformu. Zde bych doporučil SQL Server 2000 firmy Microsoft, neboť ten má datový soubor omezen na 1 TB. Dalo by se též uvažovat o profesionálních databázích typu Oracle a podobných. Pokud by to z finančních důvodů nebylo možné, dalo by se též využít služeb produktu MySQL, což je freewarovy SQL server.

Jako další nevýhodu vidím použití komunikačního protokolu DirectODBC, který zřejmě nebude stačit budoucím požadavkům na objem přenesených dat.

5.3 Náměty na další práce

Námět na další práce vidím jednoznačně v tom, pokusit se zlepšit komunikaci mezi jednotlivými moduly.
Dále by mohlo být zajímavé porovnání různých komunikačních protokolů využívaných pro přístup do databáze běžící na platformě SQL v rámci rozhraní DirectODBC (TCP/IP, IPX/SPX, pipelines, …).

Seznam použité literatury:

Habilitační práce Doc. Pavla Přibyla
Nápověda aplikace Microsoft Access 2000
Nápověda aplikace Microsoft SQL Server 2000
Vědomostní báze Microsoft MSDN Library

www.mvps.org/access/modules.html