Z V L Á Š T N O S T I P R O G R A M O V Á N Í
P R O J E D N O T L I V É K O M P O N E N T Y
D A T A B A N K O V É H O S Y S T É M U

Zbyněk Štěpánek, prom.mat.

Abstrakt: Databankové systémy bývají v literatuře popisovány převážně z hlediska informačních systémů. Méně pozornosti bývá věnováno programové části systému a problematice tvorby uživatelských programů. Podobně jako u klasických ne-databankových souborů, i v oblasti banky dat se hledají možnosti uplatnění obecných parametrických programů. Příspěvek se zabývá problémem tvorby parametrického programu pro aktualizaci báze dat a některými zvláštnostmi uživatelských programů v dalších komponentách databankového systému /DBS/.

1. Úvod

Se vznikem a rozvojem výpočetní techniky je neodmyslitelně spjat vznik a rozvoj programování. Po krátké době závislosti programů na konkrétní technické stavbě počítače nastává /všichni si myslíme, že věčné/ období nezávislosti programů nejen na konkrétním počítači, ale víceméně i na technickém pokroku ve výpočetní technice. V současné době rychlý až překotný vývoj v technické oblasti a s tím spojený nebývalý rozmach požadavků uživatelů výpočetní techniky vede k tomu, že programátoři, kteří technologii programování propracovali do vysoce obecné úrovně, jsou často v konkrétních podmínkách tvorby aplikačních programů v nesnázích. Objevují se nové vnější podmínky programování, nové prostředí /ve smyslu filosofického Umwelt/, je třeba brát do úvahy nové faktory. Programátorské prostředí je velmi nestálé. Vytváří jej operační systémy, programovací jazyky, datové struktury, přístupy k datům, požadavky integrace a to, co se poněkud nepřesně nazývá filosofií systému. Všechny tyto složky programátorského světa jsou velice proměnné, jsou v neustálém pohybu. Snad jedinou stálou složkou je programátor sám. Není divu, že za těchto podmínek je programátor často novou situací zaskočen, a přest, že používá doporučovaných programovacích technik, jeho dílo nepůsobí přesvědčivě.

Jedním z takových úskoků na programátora je,že je postaven do databankového prostředí. V tom okamžiku je třeba mu poradit, které mezerky v programovací činnosti nabyly na důležitosti,které může přenechat na starost systému,a na které musí zapomenout. Navíc je databankový systém složen z komponentů s rozdílnými funkcemi a zaměřením,a tedy i s rozdílnými požadavky na aplikační programy. K zahození nejsou také zkušenosti s obecnými parametrickými programy,které výrazně ulehčovaly programování prací u klasických,např.sekvenčních souborů,ba programování přenechávaly přímo uživatelům - neprogramátorům. Nabízí se otázka,zda také v databankovém prostředí je možné často se opakující algoritmy řešit jediným obecným parametrickým programem. Platí to především pro bázi dat,kde se využívá velkého počtu aktualizačních algoritmů pro aktualizace dat a datových vazeb. Otázka parametrických programů a jejich uplatnění v DBS je o to zajímavější,že zde je uživatel nedělitelnou součástí systému a parametrické řízení přístupu k informacím v bázi dat jednou ze základních charakteristik banky dat.

Z nových skutečností,před které je postaven programátor při přechodu do databankového prostředí,si povšimneme

- -výskytu nových entit a jejich parametrického vyjádření
- -otevřenosti aplikačních programů vzhledem k uživateli
- -zaměření aplikačních programů podle druhu uživatele a udržovatele programů
- -funkčního rozdělení dat pro potřeby parametrické manipulace s daty.

2. Datové entity v databankovém systému

Z komponent databankového systému si v této kapitole povšimneme především báze dat a uživatelského propojení. Bázi dat definujeme jako uspořádanou dvojici - data,relace mezi daty. Obě složky mají přitom stejnou důležitost. Uživatelské propojení je programový systém,přes který vstupují do DBS parametry aktualizace báze dat a aktualizační data. Při programování aktualizace sekvenčních souborů vystupovaly jako datové entity věta,elementární položka,klíč. Pro potřeby parametrického vyjádření datových entit v DBS si zavedeme následující formální definice:

Označme p_i elementární položku ve větě v_i. Set mezi větami v_i a v_j označme s_{ij}. Postavení věty v bázi dat je odlišné od postavení věty sekvenčního souboru. Entitě "věta sekvenčního souboru" odpovídá /zde pro potřebu parametrického vyjádření datových entit/ entita cesta - PATH. Definice cesty je $C=/v_1,s_{12},v_2,\ldots$ $\ldots,s_{n-1,n},v_n/$, kde v_1 nazveme vstupní větou cesty a samozřejmě může být $v_i = v_j$ pro $i \neq j$. Dále zavedeme pojem vazby $R=/v_j,s_{j,j+1},\ldots,v_{k-1},s_{k-1,k},v_k/$ a elementární vazby $R_e=/v_j,s_{j,j+1},v_{j+1}/$. Uvedené definice se netýkají dat, ale datových representací. Samotná data se v bázi dat vyskytují jako výskyt věty $d/v_i/$, složený z výskytů elementárních položek $d/p_i/$. Pro nalesení výskytu věty $d/v_i/$ v bázi dat je třeba znát tzv. identifikační položky pro větu v_i. Označme je $I/v_i/ = /p_1,\ldots,p_n/$. Znalost $d/p_1/$ až $d/p_n/$ nám umožňuje přístup k výskytu věty $d/v_i/$. Strukturu báze dat lze tedy popsat pomocí vět v_i a setů s_{ij}, datová část báze dat je tvořena výskyty vět $d/v_i/$ a relační část je rozložitelná na elementární vazby. Dále si povšimneme dat v uživatelském propojení. Přes tuto komponentu vstupují do DBS parametry aktualizace báze dat a výskyty vět a elementárních položek, které se zúčastní aktualizace. Mezi parametry se musí objevit parametrické representace jednotlivých entit. Označme je $P/v_i/$, $P/s_{ij}/$ apod. Vstupní aktualizační položky budou uvedeny jako $A/p_i/$, případně $A/I/$ pro identifikační položky. Tyto formální popisy nám poslouží k tomu, abychom stanovili podmínky pro parametrické vyjádření průběhu aktualizace pro danou větu, případně obecně pro libovolnou cestu v bázi dat.

3. Charakteristika tvorby uživatelských programů pro bázi dat a přechod k parametricky řízené aktualizaci

Báze dat tvoří z programového hlediska nejzávažnější a nejpracnější část DBS. Zatímco v jiných částech DBS jsou k dispozici systémové programy /jako příklad lze uvést CULPRIT pro oblast výstupů z báze dat/, aktualizační programy pro bázi dat musí uživatel vytvořit sám. Obyčejně se volí ten způsob, kdy pro každou větu jsou vytvořeny programy, které umožňují zápis nového výskytu věty, změnu elementárních položek ve větě a vyloučení výskytu věty z báze dat. Zapomíná se přitom na druhou rovnocennou složku

báze dat,kterou jsou relace mezi daty. Aktualisace vazeb mezi daty bývá obyčejně začleněna do aktualizačních programů pro jednotlivé věty. Někdy je umožněna pouze vypuštěním výskytů vět z báze dat a jejich opětovným zavedením se zapojením do nových vazeb. Pro uživatele,který nezná strukturu báze dat a neví,které informace jsou uloženy ve větách,a které se realisují v relační části báze dat,je potřebný jednotný způsob zadání parametrů pro aktualizaci. Ukážeme,že parametrické zadání aktualizace relační části a zadání aktualizace elementárních položek ve větách používá těchže datových entit.

Vychásejme nejprve z případu konkrétního aktualizačního programu s pevně danou cestou $C=/v_1,s_{12},v_2,\ldots,s_{n-1,n},v_n/$. Jako vstupní data programu jsou parametry a aktualisační data z uživatelského propojení. Podle druhu aktualizace vstupují do programu následující data:

a/ zápis nového výskytu věty - $I_j,A/v_j/$
b/ změna elemnt.položek věty - $I_j,A/p_j/$ nebo $A/I_j/$
c/ vypuštění výskytu věty - I_j.

Jedná se tedy vesměs o elementární datové položky,které jsou pouze vhodně seskupeny pro potřebu identifikace věty - seznam položek I_j,nebo tvoří nově zaváděná data do báze dat - $A/p_j/$ atd. Tím je formálně popsán případ aktualisace elementárních položek nebo celých vět. Dále proberme případ,kdy požadovaná změna v bázi dat neznamená změnu žádné elementární položky,ale pouze změnu vazby nebo elementární vazby mezi výskyty vět. Pro aktualisaci elementární vazby $V=/v_j,s_{j,j+1},v_{j+1}/$ stačí zadat $I_j,I_{j+1},A/I_j/$, $A/I_{j+1}/$,tedy identifikovat existující výskyty vět v_j a v_{j+1} a určit jejich nové identifikace. Z existence setu $s_{j,j+1}$,který je konstantně určen algoritmem programu vyplývá,že se provede nové propojení mezi aktualisovanými výskyty vět. Podobně tomu bude při aktualizaci vazby $V=/v_j,s_{j,j+1},\ldots,s_{k-1,k},v_k/$,kdy se při změně identifikace věty v_m změní vazby dané sety $s_{m-1,m}$ a $s_{m,m+1}$. Pokud bude $A/I_m/ = I_m$,ke změnám nedojde. I v případě aktualisace relací mezi daty vystačíme na vstupu s identifikacemi položek - seznamy I a s výskyty položek - aktualizované hodnoty A. Uživatel tedy nerozliší,zda zadává aktualisaci vět nebo relací .

Algoritmus aktualizačního programu pracuje kromě datových entit ještě s dalšími entitami. Jako příklad můžeme uvést chybové stavy /ERROR STATUS/. Při přístupu k větě v_1 a k větě v_{j+1} po setu $s_{j,j+1}$ jsou dány podmínky, jejichž logické vyhodnocení ovlivňuje další postup po dané cestě. Indikace splnění těchto podmínek jsou uvedené chybové stavy. Pro set $s_{j,j+1}$ je $E_{j,j+1}$ seznam, identifikující příslušné chybové stavy. Programátor se na ně může odvolat a rozhodovat o dalším postupu aktualizace. Parametrické vyjádření chybových stavů vyžaduje vytvoření parametrických representací pro jednotlivé chybové stavy. Uživatel samozřejmě nezná seznamy $E_{j,j+1}$. Korektní použití parametrů chybových stavů musí být zabezpečeno systémem při dialogu s uživatelem.

Dále si povšimneme případu, kdy cesta C není dána v programu konstantně, ale kdy ji musí uživatel zadat parametricky. Je to případ obecného aktualizačního programu. Vedle parametrů pro vyhledání příslušných položek a vět musí uživatel určit parametricky cestu. Pro zadání cesty C zadá vstupní větu v_1 a další věty v_2 až v_n. Příslušné sety $s_{j,j+1}$ se doplní z popisu subschematu v SŘBD. Jsou-li mezi větami v_j a v_{j+1} sety $s_{j,j+1}$ a $s'_{j,j+1}$, lze požadovaný set identifikovat z identifikací I_j a I_{j+1} a subschemového pohledu. Vlastní parametrické zadání cesty je svým součástí uživatelského propojení a bude probráno dále.

Dosud uváděné úvahy slouží jako příklad formálního vyjádření některých funkcí aktualizačního programu a vlastností datových entit. Jsou nezbytné pro sestavení algoritmu obecného parametricky řízeného programu. Kromě toho musíme brát v úvahu postavení báze dat v DBS a z toho vyplývající rozhodující charakteristiky programů v oblasti přístupu do báze dat. Programátor musí při tvorbě programů pro aktualizace báze dat počítat s tím, že

- jediným vstupem dat z uživatelského propojení je umožněna jejich jednotná struktura
- je nezbytné spojení programu se SŘBD
- obecné parametrické programy je třeba posuzovat nejen z hlediska typizace algoritmů, ale i z hlediska jejich náročnosti na systémové prostředky a strojový čas
- průběh aktualizace báze se musí podřídit celkovému zaměření DBS /interaktivní přístup uživatele apod./.

4. Určování parametrů aktualizace báze dat v dialogu mezi uživatelem a systémem řízení báze dat

Vycházejme z předpokladu, že daný DBS obsahuje technické i programové vybavení, zaručující uživateli komunikaci se SŘBD v režimu "dotaz - odpověď". Hledáme podmínky, které musí příslušný dialogový program splňovat, aby zajistil možnost parametrického zadání průběhu aktualizace báze dat, tedy vstup parametrů pro obecný aktualizační program. V první řadě musí při dialogu uživatel zadat cestu přístupu k položkám, jichž se bude aktualizace týkat. Jsou možné dva postupy. V prvním případě uživatel stanoví větu v_j obsahující aktualizovanou položku p_k. Systém v odpovědi vypíše seznam $I_j = /p_1,\ldots,p_s/$ identifikačních položek pro větu v_j. Položky $p_1,\ldots,p_s$ jsou obsaženy ve větách $v_1,\ldots,v_s$. K těmto větám opět existují seznamy jejich identifikačních položek. Po n krocích budou všechny identifikační položky $p_{1n},\ldots,p_{sn}$ obsaženy ve větách $v_{1n},\ldots,v_{sn}$. Nebude tedy třeba určovat identifikace dalších vět. Ze sítě mezi větami $v_{1n},\ldots,v_{sn}$, které jsou určeny seznamy identifikačních položek $I_{1n},\ldots,I_{sn}$ je sestavena cesta. SŘBD v tomto případě sám určí cestu i potřebné identifikační položky, a to pouze na základě stanovení výsledné aktualizované položky. Druhý způsob začíná tím, že uživatel volí vstupní větu v_1. Systém odpovídá seznamem položek, které lze aktualizovat na základě identifikačních položek z věty v_1. Současně nabídne uživateli věty, které mohou být jako v_2 pokračováním cesty. Pokud uživatel v nabídnutém seznamu požadovanou položku nenalezl, volí v_2. Krok se opakuje pro tuto větu. Po n krocích uživatel nalezne hledanou položku a cestu ukončí nebo systém další pokračování cesty nepřipouští /množina pokračovacích vět je prázdná/ dříve, než uživatel měl k dispozici hledanou položku. Cesta je tedy definována nebo byla naopak špatně volena vstupní věta v_1 a uživatel musí volit jinou. Komplikace mohou nastat v případě, že položka p se vyskytuje ve větách v_m a v_{m+1}. Při postupu podle první možnosti, tedy od koncové věty cesty, uživatel dostane cestu pro aktualizaci položky p ve větě v_{m+1}. Při druhém způsobu bude cesta začínat nějakou vstupní větou v_1 a končit větou v_m. Provede se aktualizace položky p ve větě v_m. Je-li mezi větami v_m a v_{m+1} set $s_{m,m+1}$, musí být

položka p současně aktualizována ve v_n i v_{n+1}. Tento případ musí být ošetřen v algoritmu parametrického programu, protože uživatel není v obecném případě seznámen s redundancí dat v bázi dat. Po parametrickém zadání cesty pokračuje dialog uživatele a SŘBD určením dalších datových entit.

Uvedený příklad ukazuje, jak dialogový program respektuje požadavky svého výstupu, kterým je parametrické zadání pro obecný aktualizační program. Na druhou stranu je však dialogový program otevřen k uživateli. O otevřenosti programu k uživateli mluvíme v tom případě, kdy algoritmus programu je přímo ovlivňován uživatelem v průběhu vykonávání programu. Rozlišujeme tedy programy

- uzavřené, kdy algoritmus programu probíhá podle datových souborů bez dalšího vlivu uživatele
- polouzavřené, u nichž má uživatel možnost v jistých krocích algoritmu zvolit další postup vykonávání programu
- otevřené, které jsou rozčleněny do kroků a každý krok je závislý na zásahu uživatele.

Jestliže z pohledu vnitřní struktury DBS na programy bylo potřebné definovat formální pravidla pro práci s datovými etitami, potom z hlediska uživatele musí programy zohledňovat charakteristické vlastnosti vnějšku DBS. Konkrétně v uživatelském propojení musí programy nejen zabezpečovat přenos všech potřebných informací pro aktualizační programy, ale musí být hlavně přizpůsoben odborné i psychické stránce uživatele. Zde samozřejmě opustíme metodu formálního popisu a provedeme klasifikaci uživatelů z hlediska DBS. Mnohé vlastnosti uživatelů se dají popsat pouze přibližně /modernista by řekl, že jsou mlhavé/, což je dáno tím, že databázové prostředí je nové nejen pro programátora, ale i pro uživatele. Jako příklad mohou posloužit skutečnosti, že uživatel v průběhu dialogu se SŘBD potřebuje

- podrobnější vysvětlení k dotazu systému
- ověřit si správnost vlastní odpovědi
- v kterémkoliv okamžiku dialogu poradit další správný postup.

Program uzavřený k uživateli vystačil s ošetřením správnosti vstupních dat. Program otevřený, jehož příkladem je dialogový program, musí ošetřit jakékoliv vstupní parametry zadané uživatelem a to nejen jejich syntaktickou stránku, ale i jejich neú-

plnost,nepřesnost i úmyslně nesprávnou sémantiku. Důležité je také znát typ uživatele,kterému je otevřený program určen. Z pohledu programátora rozlišujeme uživatele finálního,intermediálního a systémového. U finálního uživatele,který je nejčastější osobou při využívání dialogu s DBS,předpokládáme podrobnou znalost problematiky reality,kterou data v daném DBS zobrazují. Není však obeznámen se strukturou DBS a s uložením dat v bázi dat. Protože finální uživatel projevuje největší nerozhodnost při dialogu v uživatelském propojení,musí mu být přizpůsobena forma dialogového jazyka. Z možných forem /forma klíčových slov,forma oddělovačů,forma s pevnými místy/ volíme takovou,která na jedné straně koriguje neurčitost odpovědí uživatele,způsobenou různým pochopením významu jednotlivých entit DBS,na druhé straně neodrazuje uživatele jemu těžko pochopitelným dodržováním formalizujících pravidel dialogu. V praxi se osvědčila forma klíčových slov a forma jazyka s pevnými místy.

Poněkud jiná je situace u intermediálního uživatele. V případě,že finální uživatel není schopen vyjádřit svůj požadavek na přístup do báze dat ve formě požadované dialogovým programem,obrátí se na intermediálního uživatele. Ten je obeznámen s uložením dat v bázi dat a s tvorbou parametrů pro aktualizační program. Požadavek finálního uživatele pak realizuje přes dialog v uživatelském propojení nebo přímo sestaví vstupní data a parametry pro aktualizační program v oblasti báze dat. Posledně uvedený způsob se používá také v případě,kdy DBS není vybaven prostředky pro vedení dialogu mezi uživatelem a SŘBD.

Systémový uživatel zná fyzické uložení dat a algoritmy programů DBS /třeba z toho titulu,že je sám programoval/. Přístup do báze dat nechť si zajistí vlastním důvtipem,pokud nechce použít uživatelského programového vybavení. Každému druhu uživatele je povolen přístup do báze dat na jiné úrovni. Finální uživatel přistupuje k datům pouze přes parametry uživatelského propojení. Intermediální uživatel má k dispozici týtéž prostředky,ale může jich efektivněji využít. Protože zná strukturu báze dat,může například zvolit optimální cestu pro aktualisaci dané věty nebo položky. Systémový uživatel má navíc k dispozici systémové programy DBS.

5. Programování pro metainformační systém

Databankový systém slouží jako informační systém pro uživatele. Uživatel si do něj vkládá informace a jiné informace z něho získává. Výhodou DBS je,že struktura báze dat umožňuje získat další informace,které do ní jako primární informace nevstoupily. Uživatel například vloží do báze dat informace realizované jako elementární vazby $/v_1,s_{12},v_2/$ a $/v_2,s_{23},v_3/$. Z vazby $/v_1,s_{12},v_2,s_{23},v_3/$ získá informaci o vztahu mezi položkami vět v_1 a v_3. Kromě těchto informací o realitě zobrazené do dat a vztahů mezi daty existují informace o DBS a jeho fungování. K danému DBS tak máme metainformační systém. Na úrovni DBS mluvíme o datech, jako výskytech elementárních položek $d/p_i/$,výskytech vět $d/v_i/$ apod. a jejich reprezentacích v popisu systému - elementární položky p_i a věty v_i. V metainformačním systému jsou data z DBS realitou, informace o těchto datech se ukládají do dat metainformačního systému - metadat. K nim zase existuje popisné vyjádření v rámci metainformačního systému. Při přechodu z primárního informačního systému /DBS/ k nadřazenému systému /metainformačnímu systému pro DBS/ zůstávají zachovány základní vztahy mezi realitou,daty se vztahem k této realitě a popisem datových entit v systému. Proto může mít metainformační systém jako své komponenty bázi dat, uživatelské propojení atd. Při tvorbě programů pro tento systém lze jejich část zaměřenou k systému budovat obdobně jako u programů v primárním informačním systému. Obecný aktualizační program proto můžeme uplatnit i při aktualizaci báze metadat. S přechodem k vyššímu informačnímu systému se podstatně změní ty programy,které jsou k uživateli otevřené. Metainformační systém získává vstupní data jednak z vlastního uživatelského propojení,kde počítá s užším okruhem uživatelů,jednak z primárního informačního systému. V DBS probíhá statistické vyhodnocování jeho fungování s propojením na met_informační systém. Uživatel DBS tedy nepřímo ovlivňuje i data v nadřazeném systému a naopak,uživatel se musí podřídit pravidlům vyplývajícím z dat tohoto systému. Ochrana a utajení dat mohou být toho příkladem. Dialogový program porovnává dotazy uživatele s kódy ochrany dat v metainformačním systému a rozhoduje o oprávněnosti požadavku uživatele na vstup do báze dat. Údaje o ochraně dat v bázi dat jsou data o datech, tedy metadata a patří proto do metainformačního systému.

6. Estetická úvaha na závěr

V předcházejících odstavcích jsme si povšimli některých stránek programování uživatelských programů v databankovém prostředí. Nejprve to byla ukázka formálního vyjádření datových entit a práce s nimi pro potřeby parametrických programů. Dále jsme probrali otázku zohlednění typu uživatele v dialogových programech. Programy mohou splňovat požadavky obou stran - jak systému,do kterého jsou začleněny,tak uživatele,který je využívá - a přece nemusí být hojně používány. Hlavně se to týká obecných parametrických programů,které jsou určeny k rozšíření mezi značný počet uživatelů. Program musí mít ještě jistou estetickou hodnotu. K používání se program dostane přes svého udržovatele,který se postaral o jeho zařazení do programového vybavení systému a sleduje jeho správnou funkci. Udržovatel při posuzování daného programu bere v úvahu hlediska ekonomické,komerční,funkční a estetické. Estetická hodnota programu je tvořena

- dokonalostí formální stránky
- použitím programových prostředků úměrných úrovni systému
- sladěním výrazových prostředků uživatele a výrazových prostředků programu
- jasnou základní orientací na problémový okruh uživatele.

Estetika programů má však ještě daleko do vědecké povahy. Konečně ani estetika uměleckých oborů není považována za vědu,ale za nauku. A to je znám případ zastřeleného klavíristy pro špatnou hru /na Divokém západě/, případ zastřeleného programátora kvůli špatnému programu však znám není.