LMD mokslo darbai, III tomas, 217–221

Dalykinių programų interfeisas su deduktyvine duomenų baze

Albertas ČAPLINSKAS (MII), Mindaugas MILAŠAUSKAS (VGTU)
el. paštas: alcapl@ktl.mii.lt, mind@lema.lt

Įvadas

Straipsnyje nagrinėjama vadinamoji tarpinė programinė įranga (angl. *midleware*), skirta dalykinių programų darbo su deduktyvinėmis duomenų bazėmis kurti. Straipsnio tikslas – pasiūlyti naują tokios programinės įrangos organizavimo būdą.

Dalykinių programų interfeisai

Tradicinėse programų sistemose programos ir duomenų bazių sąveikos problema buvo sprendžiama „panardinant" duomenų manipuliavimo kalbas (DML) į programavimo kalbas [1]. Kitaip tariant, programavimo kalba būdavo papildoma operatoriais, leidžiančiais manipuliuoti duomenų bazėje saugomais duomenimis taip, tarsi jie būtų vidiniai programos duomenys.

Šiuolaikinės programų sistemos dažniausiai esti išskirstytos ir veikia heterogeninėje aplinkoje. Heterogeninės aplinkos elementai nuolat keičiami, todėl sistemų komponentai turi būti rašomi taip, kad jie galėtų naudotis kitais komponentais, nežinodami tikslių tų komponentų charakteristikų. Todėl programa negali dirbti su duomenų baze DML priemonėmis, nes jos kiekvienai DBVS yra specifinės. Komponentų sąveika heterogeninėje aplinkoje paprastai realizuojama naudojant tarpinę programinę įrangą, kurios priemonėmis sukuriami standartizuoti dalykinių programų interfeisai (angl. *application program interface* (API)).

Dalykinių programų interfeisai esti kelių lygmenų. Interfeisų hierarchijos kuriamos taip, kad išskirstyta programų sistema taptų koncepciškai skaidri ir ją sudarančioms programoms, ir ja besinaudojantiems žmonėms. Kitaip tariant, tarpinė programa turi paslėpti visas heterogeninės aplinkos detales ir leisti rašyti programas bei naudotis jomis taip, tarsi visa išskirstyta programų sistema veiktų viename kompiuteryje.

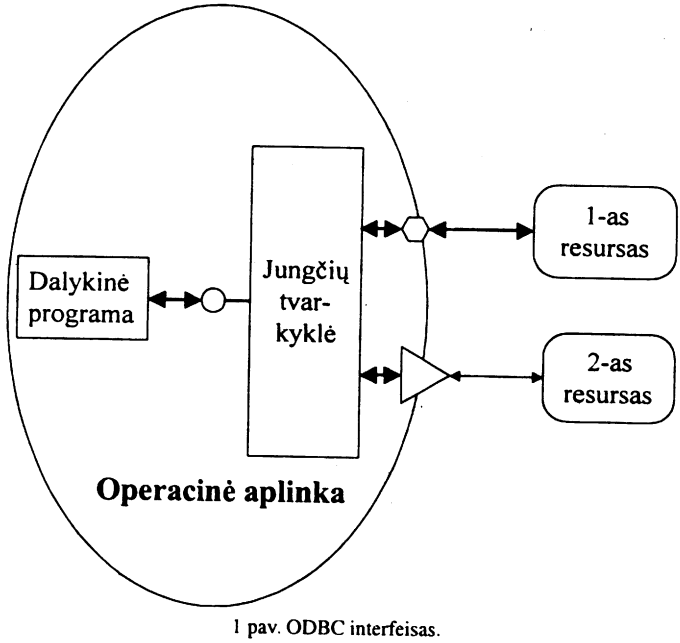
ODBC interfeisas ir jo raida

Dirbant su duomenų bazėmis, tarpinė programinė įranga turi sukurti interfeisą, leidžiantį formuluoti užklausas duomenų bazei, nežinant kokios duomenų bazių valdymo sistemos priemonėmis ta bazė yra realizuota. Reliacinėms duomenų bazėms užklausos paprastai formuluojamos SQL

kalba [2]. Viena iš populiariausų priemonių SQL interfeisui kurti yra vadinamoji ODBC (angl. *open database connectivity*). Tai firmos *Microsoft* sukurta tarpinė programinė įranga. Ji organizuota kaip specialios tvarkyklės (angl. *manager*) tvarkoma duomenų bazių jungčių (angl. *driver*) biblioteka. Jungtys jungia dalykines programas su skirtingų DBVS priemonėmis tvarkomomis duomenų bazėmis. Dalykinė programa pateikia ODBC užklausą SQL kalba. Kokia konkreti duomenų bazės jungtis bus panaudota bendruoju atveju programuotojui žinoti yra nebūtina. Dalykinė programa su atitinkama jungtimi susiejama specialiai tam skirtu aprašu.

Dalykinė programa naudojasi ODBC kaip vienu iš *MS Windows* dalykinių programų interfeisų (angl. API). Kitaip tariant, dirbant su ODBC naudojamasi specialiai tam skirtų funkcijų rinkiniu. Numatytos specialios funkcijos pradėti darbą su duomenų baze ir jį baigti, darbo su duomenų baze sesijai valdyti, užklausoms pateikti, klaidoms apdoroti ir kt..

ODBC grindžiama koersijos principu. Kitaip tariant, bet kuri duomenų bazės jungtis šiuo atveju atlieka du vaidmenis. Viena vertus, ji veikia kaip preprocesorius, transformuojantis užklausą iš standartinio pavidalo į vidinį atitinkamos duomenų bazių valdymo sistemos formatą, antra vertus, kaip postprocesorius, transformuojantis užklausos rezultatus į dalykinės programos duomenų struktūras.



1 pav. ODBC interfeisas.

Šitoks tarpinės programinės įrangos organizavimo būdas turi keletą privalumų. Svarbiausieji iš jų yra tai, kad yra gana paprasta veikiančioje programoje pakeisti vieną DBVS kita ir kad, sukūrus naują reliacinę duomenų bazių valdymo sistemą, jos priemonėmis tvarkomas bazes galima padaryti prieinamas dalykinei programai paprasčiausiai papildžius ODBC biblioteką nauja duomenų bazės jungtimi. Kita vertus, šitoks tarpinės programinės įrangos organizavimo būdas

turi taip pat ir rimtų trūkumų. Visų pirma šitokia tarpinė programinė įranga yra nepakankamai naši. Antra, jungčių bibliotekos konfiguracijos tvarkymas yra gana sudėtingas. Tačiau rimčiausias trūkumas yra tai, kad daugeliu atveju, pavyzdžiui, programuojant Microsoft Visual Basic ar Microsoft Access priemonėmis, nepatogu naudotis MS Windows lygmens dalykinių programų interfeisu. Toks semantinis lygmuo yra per daug žemas. Problemą bandoma spręsti naudojantis ODBC per vadinamąją *Jet duomenų bazių mašiną* (specialų prieities prie duomenų bazių objektą), bet šitaip praranda dalis ODBC funkcionalumo.

Siekiant pašalinti šiuos trūkumus, buvo sukurti du aukšto semantinio lygmens interfeisai – DAO (angl. *data accsess objects*) ir RDO (angl. *remote access objects*). Naudojantis šiais interfeisais galima paprastai pasinaudoti ODBC ir prieiti prie reliacinių duomenų bazių iš programų, rašomų aukšto semantinio lygmens programavimo kalbomis. Tačiau tam, kad intensyviai dirbti su Interneto ir kitomis nereliacinėmis duomenų bazėmis, DAO ir RDO interfeisų teikiamų galimybių nepakanka. Tam reikia tarpinės programinės įrangos, kuriančios tokį interfeisą, per kurį būtų galima dirbti ne tik su reliacinėmis duomenų bazėmis, bet ir su bet kuriuo kitu duomenų šaltiniu. Siekdama išspręsti šią problemą, firma Microsoft sukūrė vadinamąją universalios prieities prie duomenų strategiją (angl. *strategy for universal data access*) [3], [4], esminiai išplečiančią ODBC, DAO ir RDO funkcionalumą. Strategija grindžiama objekto komponento modeliu. Ji realizuota vadinamąja OLE DB. OLE DB koncepcija numato, kad kiekvienas duomenų šaltinis turi specialų duomenų teikėją (angl. *data provider*). Šitaip vadinami objektai, realizuojantys prieitį prie to šaltinio duomenų. Duomenų teikėjai realizuojami COM objektų rinkiniais. Kiekvienas duomenų teikėjas pateikia dalykinei programai standartizuotų interfeisų rinkinį, leidžiantį naudotis pačiomis įvairiausiomis DBVS paslaugomis (transakcijos, eilių tvarkymas, apsauga ir kt.). Kadangi duomenų teikėjai yra COM objektai, jais galima naudotis, rašant programas dauguma šiuo metu vartojamų programavimo kalbų (C++, Java, Basic ir kt.). Programuotojas dirba su duomenų teikėjais per vadinamąjį ADO (angl. *ActiveX data objects*) interfeisą. Duomenų teikėjai grąžina rezultatus programoms lentelės pavidalu.

Interfeisas deduktyvinei DB

Universaliosios prieities prie duomenų strategijos principai gali būti panaudoti taip pat ir prieičiai prie deduktyvinių duomenų bazių realizuoti. Vienok, mūsų nuomone, šitoks darbo su deduktyvinėmis duomenų bazėmis būdas turi keletą rimtų trūkumų. Visų pirma duomenų teikėjai pateikia rezultatus lentelės forma. Antra, programa dirba su duomenų baze pateikdama jai atitinkamas užklausas ir apdorodama tų užklausų rezultatus. Deduktyvinėms duomenų bazėms užklausų kalbos standarto kol kas nėra. Be to, naudojant šitokį darbo su deduktyvine duomenų baze būdą, dalykinėje programoje ši bazė pavaizduojama „plokščiai", nes interfeisas ignoruoja deduktyvinės duomenų bazės kalbos semantinius ypatumus ir atlieka vien tik atitinkamus sintaksinius perdirbimus. Todėl šitaip parašyta dalykinė programa yra nevienalytė, jos tekstas yra sudėtingas ir sunkiai suvokiamas.

Mūsų nuomone, tais atvejais, kuomet ir deduktyvinė duomenė bazė, ir programavimo kalba yra objektinės, auksčiau aptartus trūkumus galima pašalinti organizuojant tarpinę programinę

įrangą kitu būdu. Mes siūlome pasinaudoti OMG (angl. *Object Management Group*) objekto modelio [5] idėjomis. Pasiūlymų esmė šitokia:

- sukurti komponentą (IDL transliatoriaus atitikmenį), kuris programoje dinamiškai kurtų pseudoobjektus, atstovaujančius duomenų bazėje saugomus objektus ir matomus programoje taip, tarsi tai būtų vidiniai jos objektai,
- panaudojant standartines programavimo kalbos išplečiamumo priemones (klasės, funkcijos ir pan.), papildyti programavimo kalbą užklausos konstrukcija, leidžiant naudotis ta konstrukcija tik dirbant su duomenų bazėje saugomais objektais,
- įvesti specialų komponentą (brokerio atitikmenį), organizuojantį objektų atstovų ir duomenų bazėje saugomų objektų sąveiką.

Naudojant šitokį interfeisą, kiekvienas duomenų bazės objektas turi programoje savo atstovą (pseudoobjektą) programoje. Programuotojo požiūriu, dalykinės programos interfeisą sudaro pseudoobjektų visuma. Manipuliuojant pseudoobjektais, iš tiesų yra manipuliuojama duomenų bazės objektais ir, atvirkščiai, visi duomenų bazės objektų pokyčiai tiesiogiai atspindimi jų atstovams. Tam naudojamas delegavimo mechanizmas. Interfeisas paslepia konkrečios deduktyvinės duomenų bazės ypatumus, kitaip tariant, padaro bazę skaidria atitinkama programavimo kalba programuojančiam programuotojui. Pavyzdžiui, visos objektinės deduktyvinės duomenų bazės savybės iškeliamos į C++ lygmenį objektų klasių ir jų savybių pavidalu.

Siūlomas interfeiso su deduktyvinėmis duomenų bazėmis įgyvendinimo būdas išbandytas organizuojant C++ kalba parašytų programų sąveikai su freimų logikos [6] pagrindu realizuota deduktyvine duomenų baze *Florid*.

Išvados. Išskirstytose heterogeninėse programų sistemose dalykinių programų interfeisai darbui su duomenų bazėmis kuriami specialiai tam skirtos tarpinės įrangos priemonėmis. Šiuo metu vyrauja tendencija tarpinę programinę įrangą kurti taip, kad programa dirbtų su duomenų baze, pateikdama jai užklausų kalba formuluojamas užklausas. Rašant su (objektinėmis) deduktyvinėmis duomenų bazėmis dirbti skirtas programas objektinėmis programavimo kalbomis, dalykinės programos ir duomenų bazės interfeisą tikslingiau organizuoti sukuriant programoje duomenų bazėje esantiems objektams atstovaujančius pseudoobjektus ir leidžiant manipuliuoti tais pseudoobjektais vadovaujantis tomis pačiomis taisyklėmis, kaip ir manipuliuojant vidiniais programos objektais.

Literatūra

[1] J. Mylopoulos, P.A. Bernstein, H.K.T. Wong, Language facilities for designing database-intensive applications, *ACM Transactions on Database Systems*, 2(5), 185–207 (1980).

[2] S. Vang, *SQL and Relational Databases*, Microtrend™ Books, San Marcos, California (1991).

[3] R. Fisher, *OLE DB Provider Templates*, White Paper, Microsoft Corporation (1997).

[4] D. Lazar, *Microsoft Strategy for Universal Data Accsess*, White Paper, Microsoft Corporation, October (1997).

[5] S. Vinoski, CORBA: integrating diverse applications within distributed heterogeneous environments, *IEEE Communications Magazine*, 2(35), 35–47 (1992).

[6] M. Kifer, G. Lausen, J. Wu, *Logical Foundations of Object-Oriented and Frame-Based Languages*, Technical Report 93/06, Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794 (1993).

Application programme interface to deductive database

A. Čaplinskas, M. Milašauskas

The paper discusses midleware that implements application programme interface to deductive database. New approach is proposed how to implement the application programme interface of such kind.