Milan Gornik

LISP Query System

LISP Query System obuhvata radno okruženje i novi upitni jezik namenjen upravljanju relacionim bazama podataka. Ovo okruženje predstavlja delimičnu implementaciju sistema za upravljanje relacionim bazama podataka, a izrađeno je sa namerom da predstavi osobine upitnog jezika zasnovanog na klasičnom LISP funkcionalnom programskom jeziku. Mogućnosti definisanja podataka i manipulacije njima svrstavaju LQS upitni jezik u Data Definition Language i Data Manipulation Language klase jezika. Sistem automatski održava integritet baze podataka: integritet tipa i domena atributa, integritet primarnog ključa i referencijalni integritet. Podržane su osnovne operacije relacione algebre: unija, selekcija, projekcija i pridruživanje, kao i proizvoljne transakcije podataka u bazi. Omogućeno je dodavanje novih tipova domena i naknadna promena uspostavljenih ograničenja. Automatizovanje procesa izdvajanja željenih informacija na dinamičkom i, za korisnika transparentnom, nivou obavljaju dinamički pogledi. Mehanizmi fizičkog sloja sistema za upravljanje relacionim bazama podataka nisu značajni za konstituisanje upitnog jezika, zato su implementirani u svojim minimalnim formama.

***Ključne reči**. LISP: interpreter, ekstenzije; relacione baze podataka; upitni jezici*

Uvod

Važnu ulogu u sistemu za upravljanje relacionim bazama podataka ima njegov korisnički interfejs. U složenom i svestranom okruženju korisnički interfejs postavlja specifične implementacione zahteve pred projektanta sistema. Pored komunikacije sa krajnjim korisnikom, on obezbeđuje i rad *shell* skripta, kao i vezu ka spoljnim aplikacijama. Postoje dva metoda ostvarivanja interakcije između korisnika baze i sistema za upravljanje (pri čemu korisnikom nazivamo krajnjeg korisnika i *client* aplikaciju). Prvi se svodi na korišćenje upitnog jezika (klasičan način upravljanja), dok drugi koristi vizuelne alate u istu svrhu.

Milan Gornik (1981), Crvenka, Ive Lole Ribara 64, učenik 2. razreda Srednje tehničke škole u Somboru

U dosadašnjem razvoju upitnih jezika insistiralo se na stalnom približavanju prirodnim jezicima, što je rezultovalo posledicama kao što su: usložnjavanje sintakse, udaljavanje korisnika od realne organizacije informacionog sistema i smanjenje simetričnosti (odnosno doslednosti) tih jezika. Kvalitet ovih jezika teško je oceniti jer su navedene negativne posledice gotovo uvek prisutne. Standardni predstavnik upitnih jezika jeste SQL (Structured Query Language), koji je našao izuzetno veliku primenu u savremenim RDBMS (relational database managment system – sistem za upravljanje relacionim bazama podataka).

Vizuelni alati koriste dve tehnike: *query by example* i *query by form*. Oba principa podrazumevaju navođenje vizuelnog, tabelarnog upita od strane korisnika. Mnogi sistemi podržavaju ovakav vid komunikacije, ali se on, kao opciona nadgradnja, opet svodi na prevođenje zadanih upita u klasičan upitni jezik.

Funkcionalni programski jezici poseduju mnoge osobine koje pogoduju upotrebi u oblasti relacionih baza podataka i upitnih jezika. LISP Query System predstavlja model (delimičnu implementaciju) RDBMS, zasnovan na LISP-u, kao jeziku kojeg većina karakteristika svrstava u grupu funkcionalnih jezika. LQS integriše relacioni model podataka u LISP i omogućava upotrebu LISP-a za upravljanje tim modelom.

Koncepcija LQS

Težnja za potpunim integrisanjem u standardni LISP prouzrokovala je izradu sistema kao ekstenzije, u vidu biblioteke funkcija, postojećem LISP interpreteru. Sintaksa LQS upitnog jezika oslikava osobine bazičnog jezika; karakterišu je simetričnost, konciznost, jednoznačnost i jasan kontakt sa podacima. Simetričnost zastupa dosledno sprovođenje sintaksičkih pravila jezika, u svim situacijama i konstrukcijama. Konciznost i jednoznačnost govore o formalnoj strani jezika – o primeni minimalnog skupa funkcija, kojim se mogu obaviti sve operacije. Jasan kontakt sa podacima podrazumeva informisanost korisnika o tome kojim tipovima podataka manipuliše i koji su rezultujući tipovi; naravno, korisnik ne biva opterećen internim stvarima sistema, kontakt postoji u toj meri da je korisnik u svakom momentu svestan rezultata operacije.

Pomenute karakteristike izuzetno pogoduju programskom generisanju upita, ali visoki stepen formalizacije, u kombinaciji sa specifičnom LISP sintaksom može značajno otežati rad krajnjem korisniku. U skladu sa tim, sistem je rađen u dva sloja, gde prvi predstavlja operativni deo programa (namenjen povezivanju sistema sa drugim aplikacijama), a drugi deo se samo nadovezuje, nadograđujući interfejs na nivo pristupačniji krajnjem korisniku. Izrada rutina koje su vezane za fizičku organizaciju informa-

cionog sistema ili pak, za operativni sistem, svedena je na minimum, dok su neke funkcije RDBMS potpuno izostavljene (npr. rutine za zaštitu i oporavljanje podataka). Ograničenja postojećih LISP interpretera uslovila su nastajanje internih formata LQS-a, nekompatibilnih sa aktuelnim standardima.

Tipovi podataka

LQS se u potpunosti oslanja na prvobitnu verziju *Common LISP standarda*. Prednost LISP jezika ogleda se u visokoj simetričnosti nad jednostavnim i apstraktnim tipovima podataka. Dodatni tipovi definisani u LQS-u predstavljaju specijalizovane liste, potpuno bazirane na LISP primitivama. Postoje tri osnovna tipa podataka kojima se obavljaju gotovo sve operacije: tabele, statički pogledi i dinamički pogledi.

Tabela baze podataka nosi sledeće informacije: definiciju atributa i njihovih domena, strukturu primarnog ključa, reference na druge tabele i, naravno, konkretne podatke. Domen se određuje primenom tzv. *domain-check funkcije* (predikatske funkcije koja utvrđuje pripadnost podatka određenom domenu), pa je moguće i jednostavno kreiranje novih domena podataka, kao što je prikazano na primeru:

```
(defun magistrala (value)
   (or (member value '(isa vlb pci agp))
       (null value)))
```

Novi domen podataka, nazvan magistrala, može poprimiti vrednost nekog elementa skupa (isa, vlb, pci, agp) ili ostati nedefinisan (na NULL stanju). Na ovaj način, definisanjem domena određuju se tip i dozvoljene granice podatka, kao i mogućnost postavljanja atributa na NULL stanje.

Statički pogledi predstavljaju unekoliko modifikovane tabele. Sadrže definiciju atributa i tabelarno organizovane podatake (domeni, primarni ključ i reference ovde nisu prisutni). Moglo bi se reći da statički pogledi predstavljaju derivat tabela, odnosno način da trajno koristimo neki deo tabele (zapravo, vršimo pregled i izdvajanje željenih podataka), bez obaziranja na integritet podataka. Kreiranje statičkih pogleda obavljaju operacije relacione algebre, crpeći potrebne podatke iz polaznih tabela (zavisno od zadane operacije). Pogledima se može dalje manipulisati, ali samo onim funkcijama koje vrše novo izdvajanje podataka, bez mogućnosti da podatke i modifikujemo. Sve funkcije ovog projekta koje operišu pogledima mogu prihvatiti i tabelu kao operand, obrnuto ne važi. Podaci koje operacijama relacione algebre izdvojimo iz tabela gube svoj integritet, formirajući time statički pogled – međurezultat između polaznog skupa informacija u bazi i konačnog skupa željenih informacija.

Dinamički pogledi imaju ulogu relativno sličnu statičkim pogledima. Potpuno su kompatibilni sa statičkim pogledima, sa razlikom što ne poseduju sopstvene, permanentne podatke, već ih crpe iz ostalih pogleda ili tabela u momentu njihovog korišćenja. Uvek kada se insistira na statičkom pogledu kao parametru, možemo ravnopravno proslediti i dinamički.

Postoji još nekoliko, uslovno rečeno, tipova podataka. Oni su definisani kako bi pojedine funkcije mogle očekivati ulazne informacije formirane na tačno određeni način. Definisani su formati za opisivanje struktura novog reda podataka, primarnog ključa, kolona i spoljnog ključa tabele.

Funkcije

Prva grupa funkcija operiše tabelama baze podataka. Ove funkcije karakteriše forsiranje integriteta domena atributa i primarnog ključa, kao i referencijalnog integriteta. Osnovne funkcije ove grupe jesu funkcije za kreiranje i poništavanje tabela, i za dodavanje, odnosno, brisanje redova. Poslednje dve operacije ilustrovane su primerima:

```
(setq procesori (tadd procesori '((oznaka i80486)
    (proizvodjac Intel) (frekvencija 100))))
```

(funkcija *setq* obavlja dodelu vrednosti varijabli *procesori*; ova varijabla predstavlja tabelu baze podataka, koja će funkcijom *tadd* biti proširena novim redom podataka)

```
(setq monitori (tdelete monitori
    '(equal dijagonala 21)))
```

(na primeru je prikazano brisanje redova tabele po zadanom kriterijumu; u ovom slučaju, brišu se redovi čiji atribut *dijagonala* ima vrednost 21).

Podržane su i transakcije podataka tabela, kao i ostvarivanje referenci između pojedinih tabela:

```
(setq monitori (transac monitori
     '(cond ((equal proizvodjac 'Philips)
            (setq cena (+ cena 12.31))))))
```

(funkcija *transac* obavlja modifikaciju podataka tabele; prvo funkcijom *cond* utvrđujemo da li je potrebno obaviti modifikaciju, u ovom slučaju povećanje atributa *cena* za zadanu konstantu)

```
(bind printeri 'proizvodjac proizvodjaci)
```

(spoljni ključ tabele *printeri*, sadržan u atributu *proizvodjac* povezuje se sa primarnim ključem tabele *proizvodjaci*)

Posebne pogodnosti pri obavljanju transakcije su potpuna sloboda u formiranju izraza transakcije (svaki validan LISP izraz može biti naveden) i mogućnost modifikacije primarnog ključa tabele, pri čemu svi referencirani spoljni ključevi automatski poprimaju odgovarajuće vrednosti (metod *cascade*).

Pogledima manipulišu operacije relacione algebre: unija pogleda, projekcija kolona, selekcija redova po kriterijumu i spajanje pogleda (verzija *left outer join*). Za selekciju redova pogleda važi isto što i za transakciju podataka (proizvoljno kompleksan izraz selekcije).

Unija pogleda:

```
(setq procesori (union cpu_risc cpu_cisc))
```

Projekcija kolona pogleda:

```
(setq cenovnik (project monitori
    '(model dijagonala cena)))
```

Selekcija redova po uslovu:

```
(setq high-end (select racunari
    '(or (ram 256) (hdd 24.3))))
```

Spajanje pogleda po zajedničkim podacima:

```
(setq nabavka (join racunari
   'indeks_dilera dileri 'indeks))
```

Postoje svega dve funkcije koje operišu isključivo dinamičkim pogledima: funkcije za konstrukciju i poništavanje dinamičkih pogleda. Dinamički pogledi se potom koriste funkcijama za operisanje statičkim pogledima. Izraz kojim je određen način prikupljanja podataka u dinamički pogled predstavlja proizvoljan LISP izraz koji za rezultat ima pogled.

Rad sa bazama podataka sveden je na minimum, obezbeđujući funkcije za snimanje objekata (tabela, pogleda ili nekih drugih LISP varijabli) u datoteku baze podataka i za učitavanje ranije snimljenih podataka. Razlog minimalne implementacije ovih funkcija leži u tome što je disk I/O sistem van okvira ovog projekta.

Sledeće funkcije služe isključivo krajnjem korisniku sistema: ispitivanje tipa LQS entiteta, prikaz tabela i pogleda obe vrste u eksternom ekranskom editoru, asignacija sa zaštitom podataka i pozivanje priručnog tutorskog sistema. U prilogu ovog rada dat je spisak svih implementiranih funkcija.

Implementacija

LQS je implementiran u X-Lisp interpreteru (verzija 2.1g) predstavljajući više od 300 KB, odnosno 5700 redova izvornog koda. Uprkos tome što se ceo projekat oslanja isključivo na dokumentovane opcije prvo-

bitnog Common LISP standarda, i dalje je pitanje portabilnosti diskutabilno. Razlog tome su velike razlike između pojedinih LISP interpretera. Ipak, program je detaljno dokumentovan, tako da su sve potencijalno problematične rutine naznačene i objašnjene.

Dalji razvoj

Na sadašnjem stepenu razvoja ovog sistema ne postoje kanali za komunikaciju sa spoljnim programima. Jedina veza može se ostvariti između LQS-a i drugih aplikacija pisanih u LISP programskom jeziku (šira komunikacija zahtevala bi vezivanje za određeni operativni sistem, što bi bilo kontraproduktivno). Mogućnosti daljeg razvoja kreću se u smeru izrade kompletnog RDBMS koji bi kao upitni jezik koristio LQS jezik. Pritom, neke osobine standardnog LISP jezika morale bi biti izmenjene. Tretiranje NIL vrednosti u LISP-u jedan je od najvećih problema, jer ono nije istovetno tretmanu NULL vrednosti u relacionom modelu. Štaviše, različiti LISP interpreteri poseduju sopstvena pravila o upotrebi neodređenog NIL operanda. Razlike između simbola i stringova, kao separatnih tipova LISP-a, i poteškoće oko konverzije jednog tipa u drugi, takođe predstavljaju problem. Uvođenjem intuitivnijeg korisničkog interfejsa, vizuelno orijentisanog, problem preterane složenosti LISP izraza mogao bi biti rešen.

Zahvalnice. Sugestije Gorana Turudića predstavljale su nezamenljivu pomoć u razvoju ovog projekta. Rad je velikim delom razvijan u Istraživačkoj stanici Petnica, koja mi je ustupila neophodnu literaturu, kao i pomoć stručnih saradnika, na čelu sa Draganom Toromanom i Aleksandrom Jovičićem. Značajan udeo u finalizaciji projekta imala je i Snežana Klepić.

Literatura

Robert Wilensky, R. 1987. *LispCraft*. New York: Norton & Company

Tkalac, S. 1988. *Relacijski model podataka*. Zagreb: Informator

Alagić, S. 1985. *Relacione baze podataka*. Sarajevo: Svjetlost.

Milan Gornik

LISP Query System

LISP Query System presents new query language, which is intended for use in relation database managment. This is only a partial implementation of RDBMS (relational database managment system), developed to show some basic characteristics of a query language based on common LISP functional programming language. The ability to define and manipulate data places LQS in Data Definition and Data Manipulation language class. The system maintains database integrity: integrity of data types and domain attributes, integrity of the primary key and referential integrity. Basic relational algebra functions are supported (union, selection, project and join). It is possible to declare new domains and change already defined domains. Also, the use of dynamic views makes the process of data extraction automated and transparent for the user. Physical layer of database managment system is implemented only in it's minimal form. This layer is not crucial for building a query language.

LQS is written entirely in common LISP, as an extension of its standard, built-in functions. This ensures complete integration in LISP and provides functionality. Any standard LISP statement can be used in combination with LQS.

This project shows a new approach in query language definition and solves some of the common problems found in other script-like query languages. It's precise formal definition is more suitable for developers (or client applications) than end-users.

Future development of LQS could present visual user interface that would ease the process of database managment. Also, it will be necessarily to fully implement disk access and other low-level functions in order to build a complete RDBMS based on LISP.

Prilog: spisak funkcija

Funkcije koje manipulišu tabelama baze podataka

TCREATE, TCLOSE, kreiranje nove tabele ili poništavanje postojeće.
TADD, dodavanje novog reda podataka u tabelu.
TDELETE, brisanje redova iz tabele po određenom kriterijumu.
TRANSAC, modifikacija podataka tabele po zadanoj funkciji
BIND, UNBIND, uspostavljanje relacione zavisnosti među tabelama
DISPLAY, prikazivanje tabele u ekranskom editoru
SELECT, PROJECT, JOIN, UNION, operacije relacione algebre.

Funkcije koje operišu statičkim pogledima

SELECT, PROJECT, JOIN, UNION, operacije relacione algebre
DISPLAY, prikazivanje pogleda u ekranskom editoru.

Funkcije koje manipulišu dinamičkim pogledima

DCREATE, DCLOSE, kreiranje ili poništavanje dinamičkih pogleda
DISPLAY, prikazivanje sadržaja dinamičkog pogleda.

Dodatne funkcije

HELP, pruža dodatne informacije o korištenju LQS-a.
TSET, asignacija sa zaštitom podataka (prošireni setq).
DBSAVE, snimanje sadržaja baze podataka na disk
DBLOAD, učitavanje baze podataka sa diska.