K AKTUALIZÁCII KMEŇOVÝCH SÚBOROV

Jan VINAŘ, prom.mat., Výpočtové stredisko VS KSV, Košice

Úvod

Tento príspevok obsahuje voľný výklad algoritmu z práce [1]. Algoritmus, ktorý autor práce vypracoval nezávisle, opísal (pre sekvenčné súbory) Dijkstra [2] a za jeho autora označuje H.J. Swijena. Typické publikované algoritmy [3, 5] majú jedno spoločné: ich hlavná pozornosť sa sústreďuje na v podstate technickú otázku "chýbajúcich kľúčov"; preto riešia pridávanie a likvidáciu viet v podstate oddelene od ostatných typov opráv. Slabosť týchto algoritmov nie je v riečení problému, ale v jeho formulácii.

Formulácia problému

Čo je vlastne kmeňový súbor - a hlavne na čo existuje ? Každá veta kmeňového súboru slúži na sledovanie stavu nejasného objektu, ktorý nás zaujíma. Pre určitosť budeme hovoriť o stave bankových účtov - presnejšie ich čísel. Každú udalosť, ktorá tento stav zmení, popisujeme opravnou vetou; vlastná aktualizácia potom spočíva v zachytení vplyvu týchto zmien v kmeňovom súbore. Pre bankový účet typické zmeny budú: vytvorenie účtu, zrušenie účtu, výber a vklad (pre ilustráciu to stačí). Pri dávkovom spracovaní môže každá veta kmeňového súboru ovplyvniť rôzny počet opravných viet (jedna, viacero, alebo aj žiadna). Či sa dve opravné vety pre ten istý účet dostanú do jednej dávky alebo do rôznych dávok, závisí v podstate od náhody. Preto musíme dbať o to, aby dve opravy v jednej dávke mali ten istý účinok, ako keď ich rozdelíme. Môžeme si predstaviť napr. takúto postupnosť udalostí: zriadenie účtu 1234, vklad na účet 1234, výber účtu 1234, zrušenie účtu 1234, zriadenie nového účtu 1234, vklad na nový účet 1234 a požadovať, aby ho náš aktualizačný program správne spracoval.

Triedenie opráv

Dávkové spracovanie slúži na to, aby sme v priebehu dávky každú kmeňovú vetu čítali (a prípadne zapisovali) najviac jedenkrát. Treba preto opravné vety zotriediť podľa kľúča. V rámci kľúča je otázka triedenia zložitejšia. Z horeuvedeného príkladu vidíme, že nie je vhodné triediť ich podľa typu opravy (ako napr. v [4]). Postupnosť opravných viet musí presne odrážať poradie udalostí v reálnom svete.

Musíme teda každej opravnej vete priradiť poradové číslo a triediť v rámci kľúča podľa neho.

Stav kľúča

Tento pojem (a to, že ide o stav kľúča a nie účtu) je jadrom celého programu. Ak je číslo účtu pridelené, potom stavom tohto čísla je (aspoň) názov a zostatok účtu; inak je to informácie, že kľúč nie je pridelený. Každá reálna štruktúra kmeňového súboru je tak vlastne odrazom "ideálneho súboru", kde každej možnej hodnote kľúča odpovedá veta, obsahujúca o.i. indikátor "pridelenia kľúča".

Jadrom celého nášho programu (obr. 1) je cyklus, ktorý spracúva postupnosť kľúčov. Pre každý kľúč

(1) Zistí jeho počiatočný stav
(2) Realizuje opravy v príslušnom poradí; tým sa stav kľúča vo všeobecnosti zmení
(3) Zapíše nový stav kľúča do súboru.

Týmto pohľadom získavame niekoľko výhod:

a) Všetky otázky organizácie súboru sa sústredia v krokoch (1) a (3) cyklu.
b) Tam sa rieši aj otázka "chýbajúcich kľúčov".
c) Pridanie a zrušenie vety už nie sú špeciálne prípady; menia jednoducho stav kľúča ako každý iný typ opravnej vety.

Na obr. 2 vidíme realizáciu tohto programu v jazyku COBOL. Všimnime si niekoľké charakteristické črty:

a) Program sa zjednodušuje tým, že sa opravný súbor číta o jednu vetu dopredu [3] ; prvá veta sa číta v odstavci OTVOR-SUBORY.
b) Pre zjednodušenie výberu kľúčov z viet sa používa na príslušných miestach príkaz MOVE CORRESPONDING. Toto je jediný prípad, keď používame prostriedky, špecifické pre daný jazyk. Umožňuje to riadiť výber kľúčov deklaráciami.
c) Procedúry POCIAT-STAV a KONECNY-STAV budú špecifické pre určitú organizáciu súboru. POCIAT-STAV hľadá v kmeňovom súbore vetu s daným kľúčom. Ak ju nájde, prečíta ju do oblasti KMEN a nastaví indikátor ALLOC (kľúč je pridelený) na ANO; inak nastaví ALLOC na NIE (KMEN nehrá v tomto prípade nijakú úlohu) KONECNY-STAV testuje ALLOC; ak je ANO, zapíše alebo prepíše vetu s kľúčom ZMENY-KLUC; inak ju vyradí (ak existuje).

Poznámka: V prípade nesekvenčných súborov je vhodné, aby KONECNY-STAV mal k dispozícii aj hodnotu ALLOC, ako ju nastavil POCIAT-STAV. Tak sa môže rozhodnúť, či vetu zapísať alebo prepísať.

a) Jadrom programu je procedúra SPRACUJ-JEDEN-KLUC, ktorá sa opakuje, kým BEZNY-KLUC (t.j. kľúč práve spracovávanej opravy) nenadobudne špeciálnu hodnotu NEKONECNO, ktorá signalizuje koniec súboru opráv. Táto procedúra opakovane vyvoláva procedúru SPRACUJ-JEDNU-OPRAVU, ktorá nielen vykoná opravu, ale aj prečíta ďalšiu opravnú vetu.

Sekvenčné a nesekvenčné súbory

Na rozdiel od nesekvenčných súborov existuje len jedna organizácia sekvenčných súborov. Preto sme v tomto prípade mohli vypísať aj procedúry POCIAT-STAV a KONECNY-STAV. Ďalej v tomto prípade treba prečítať (a skopírovať) aj tie vety "starého" súboru, ktoré sa neaktualizujú. V ich prípade sa počet priebehov "cyklom opráv" rovné nule. Na obr. 3 vidíme procedúry OTVOR-SUBORY, ZATVOR-SUBORY a VYBER-DALSI-KLUC pre nesekvenčné súbory, na obr. 4 sú tie isté procedúry (spolu s procedúrami POCIAT-STAV a KONECNY-STAV) pre súbory sekvenčné.

Vlastné opravy

Procedúry pre vlastné opravy (špecifické pre danú úlohu) vidíme na obr. 5. Pozornosť zasluhujú nasledujúce fakty:

1. Tvar procedúr nezávisí od organizácie súboru.
2. Pri každej oprave môžeme (dokonca musíme) testovať indikátor ALLOC. Meníme ho iba v prípade zrušenia vety v založenia vety.

Záver

Hoci na prvý pohľad sme sa tu zaoberali veľmi zjednodušeným príkladom, pozornejší rozbor ukáže, že v rámci uvedeného algoritmu môžeme riešiť všetky aktualizačné úlohy, u ktorých logickú súvislosť medzi opravami sprostredkuje výlučne stav kľúča v kmeňovom súbore. Opísaný program (presnejšie schéma programu) je modulárny a ľahko sa ladí a udržieva. Tvorba a testovanie môžu prebiehať zhora dolu. Odkiaľ pochádzajú všetky tieto výhody ? Nejde tu o špecifickú konštrukciu programu, ale o formuláciu problému - o všeobecnú zásadu, ktorú autor [1] sformuloval takto: "Fyzická štruktúra súboru je obyčajne skresleným a optimalizovaným obrazom jednoduchej logickej štruktúry. Spracujte logickú štruktúru a optimalizačné triky skryte !"

LITERATÚRA

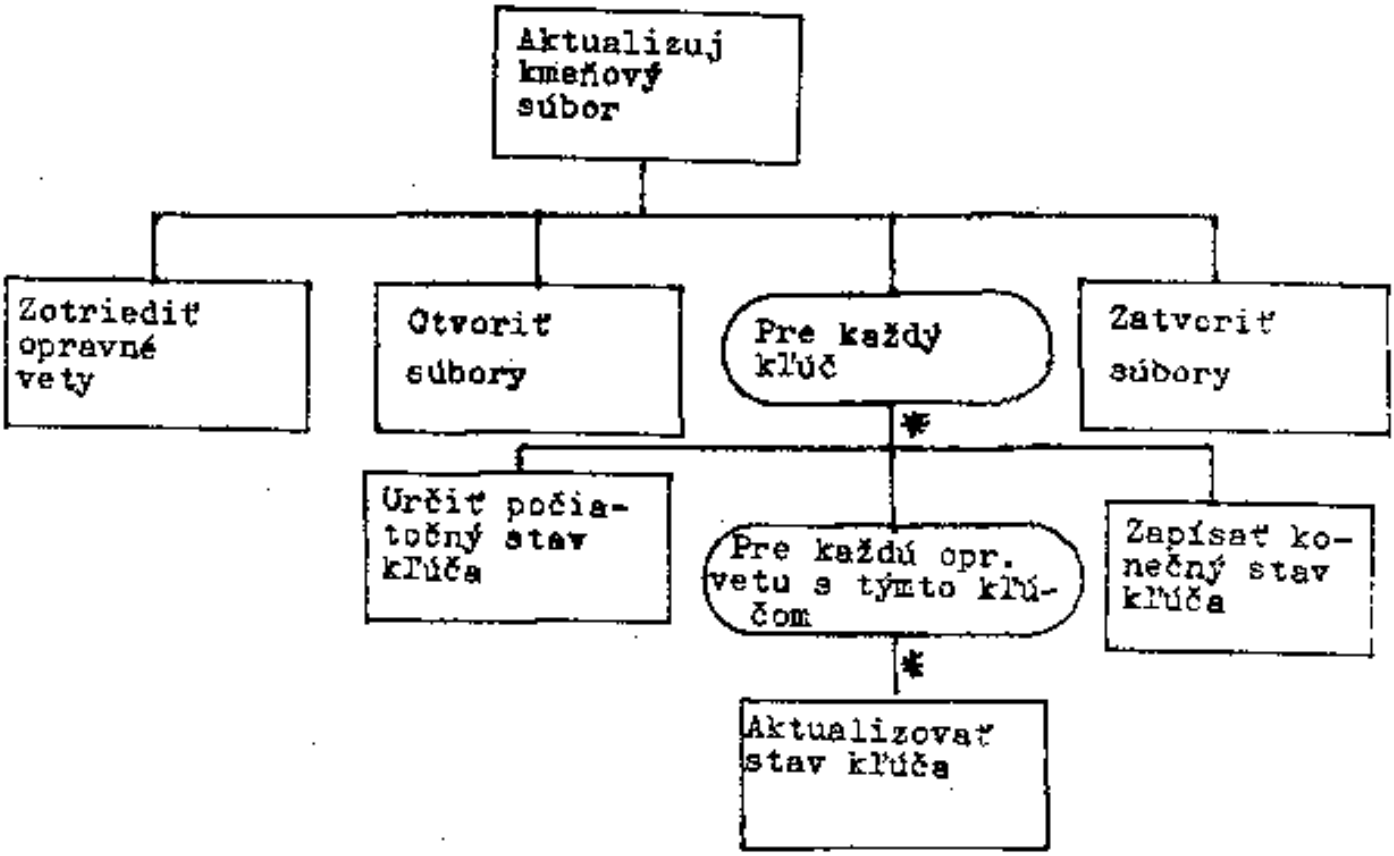
[1] B. Dwyer: One More Time - How to Update a Master File, Comm. ACM 24(1981) 1,3

[2] E. Dijkstra: A Discipline of Programming Prentice - Hall,1976

[3] M.A.Jackson: Principles of Program Design Academic Press 1975

[4] P.Vereš̆vársky: Štandardizácia programov pre údržbu bázy dát, Programování '83

[5] D.D. Mc Cracken: A Simplified Guide to Structured Cobol, Wiley 1976



Obr. 1. Štruktúra aktualizačného programu.

```
PROCEDURE DIVISION.
AKTUALIZUJ-KMENOVY-SUBOR.
  PERFORM OTVOR-SUBORY.
  PERFORM VYBER-DALSI-KLUC.
  PERFORM SPRACUJ-JEDEN-KLUC UNTIL
          BEZNY-KLUC=NEKONECNO.
  PERFORM ZATVOR-SUBORY.
  STOP RUN.
SPRACUJ-JEDEN-KLUC.
  PERFORM POCIAT-STAV.
  PERFORM SPRACUJ-JEDNU-OPRAVU
         UNTIL KLUC-OPRAVY NOT =
               BEZNY-KLUC.
  PERFORM KONECNY-STAV.
  PERFORM VYBER-DALSI-KLUC.
SPRACUJ-JEDNU-OPRAVU.
  PERFORM OPRAV-KMEN.
  PERFORM CITAJ-OPRAVU.
CITAJ-OPRAVU.
  CITAJ OPRAVNY-SUBOR AT END
        MOVE NEKONECNO TO OPRAVNY-KLUC.
        IF OPRAVNY-KLUC NOT=NEKONECNO
        MOVE CORRESPONDING OPRAVA TO OPRAVNY-KLUC
```

Obr. 2. Aktualizácia kmeňového súboru

```
OTVOR-SUBORY.
  OPEN INPUT OPRAVNY-SUBOR, I-O
  KMENOVY-SUBOR.
  PERFORM CITAJ-OPRAVU.
ZATVOR-SUBORY.
  CLOSE OPRAVNY-SUBOR,KMENOVY-SUBOR.
VYBER-DALSI-KLUC.
  MOVE OPRAVNY-KLUC TO BEZNY-KLUC.
```

Obr. 3. Procedúry pre nesekvenčný súbor

```
OTVOR-SUBORY.
  OPEN INPUT OPRAVNY-SUBOR, STARY-SUBOR
  OUTPUT NOVY-SUBOR
  PERFORM CITAJ-OPRAVU.
  PERFORM CITAJ-STARY.
ZATVOR-SUBORY.
  CLOSE OPRAVNY-SUBOR, STARY-SUBOR, NOVY-SUBOR.
VYBER-DALSI-KLUC.
  IF OPRAVNY-KLUC < STARY-KLUC
     MOVE OPRAVNY-KLUC TO BEZNY-KLUC.
     ELSE MOVE STARY-KLUC TO BEZNY-KLUC.
POCIAT-STAV.
  IF STARY-KLUC=BEZNY-KLUC
     MOVE STARY TO KMEN
     MOVE ANO TO ALLOC
     PERFORM CITAJ-STARY
  ELSE MOVE NIE TO ALLOC.
KONECNY-STAV.
  IF ALLOC=ANO WRITE KMEN.
CITAJ-STARY.
  READ STARY-SUBOR AT END
  MOVE NEKONECNO TO STARY-KLUC.
  IF STARY-KLUC NOT=NEKONECNO
  MOVE CORRESPONDENG  STARY TO STARY-KLUC.
```

Obr.4. Procedúry pre sekvenčný súbor.

```
OPRAV-KMEN.
   IF DRUH OF OPRAVA=PRIDAJ
      PERFORM PRIDAJ-VETU
   ELSE IF DRUH OF OPRAVA=ZRUS
      PERFORM ZRUS-VETU
   ELSE IF DRUH OF OPRAVA=VKLAD
      PERFORM UROB-VKLAD
   ELSE IF DRUH OF OPRAVA=VYBER
      PERFORM UROB-VYBER
   ELSE PERFORM CHYBOVE-HLASENIE.
PRIDAJ-VETU.
   IF ALLOC=ANO
     DISPLAY OPRAVA,'UCET U2 PRIPRAVENY'
     ELSE MOVE ANO TO ALLOC,
          MOVE CORRESPONDING OPRAVA
               TO KMEN,
          MOVE ZERO TO STAV OF KMEN.
ZRUS-VETU.
   IF ALLOC=NIE
      DISPLAY OPRAVA,'UCET NEEXISTUJE'
      ELSE IF STAV OF KMEN NOT ZERO
           DISPLAY OPRAVA, 'ZOSTATOK NIE JE NULA'
      ELSE MOVE NIE TO ALLOC.
UROB-VYBER.
   IF ALLOC=NIE
      DISPLAY OPRAVA, 'VYBER Z NEEXISTUJUCEHO UCTU'
      ELSE SUBTRACT SUMA OF OPRAVA
           FROM STAV OF KMEN.
UROB-VKLAD.
   IF ALLOC=NIE
      DISPLAY OPRAVA, 'VKLAD NA NEEXISTUJUCI UCET'
      ELSE ADD SUMA OF OPRAVA
           TO STAV OF KMEN.
CHYBOVE-HLASENIE.
   DISPLAY OPRAVA, 'NEZNAMY ZMENOVY KOD'
```

Obr. 5. Vzorové procedúry pre opravu.