Jan VINAŘ, prom. mat.
Výpočtové stredisko VSl.KNV Košice

TRIEDENIE STREDNE VEĽKÝCH SUBOROV

Úvod

Aj v dnešnej dobe databankových systémov, virtuálnej pamäte a iných vymožeností, je triedenie sekvenčných súborov stále dôležitým prvkom práce programov. Obsahom tohto príspevku je popis techniky, ktorá môže v niektorých prípadoch nahradiť vonkajšie triedenie a prináša niektoré výhody.

Programovaciu techniku je dosť ťažké opísať všeobecne. Zvolil som preto konkrétnu situáciu, v ktorej sme ju použili, "očistenú" od technických detailov, a pokúsim sa celý postup zlúčiť tak, aby si pozorný poslucháč mohol základné myšlienky preniesť na svoj problém.

Situácia

V systéme spracovania účtovníctva pre určitú skupinu organizácií je základnou jednotkou informácie účtovný zápis. Po určitom predbežnom spracovaní ho zobrazuje veta pohybov (obr. 1). Po zotriedení podľa triediacich kľúčov získavame súbor pohybov (obr. 2), ktorý je hlavným vstupom do systému. Výstupom sú rôzne typy zostáv, ktoré sa spracujú podľa organizácií. Triedenie viet pre každú zostavu je odlišné.

Pokusy o riešenie

1. Ako sa systém vyvíjal z jednotlivých programov, vytváral sa pre každú zostavu osobitný stavový súbor. Všetky tieto súbory sa aktualizovali jedným pohybovým súborom. Počet súborov čoskoro presiahol hranicu zvládnuteľnosti.
2. Pre všetky zostavy sa vytvoril jediný stavový súbor. Ten sa pred spracovaním vždy zotriedil pomocou systémového triediaceho programu. To ovšem drasticky obmedzovalo rozsah súboru.
3. (Teplejšie) Využívame to, že podľa organizácií je náš súbor (stavový či pohybový) už zotriedený (obr. 2). Pri spracovaní jednej organizácie postupujeme preto takto:
   a) načítame na disk vety za danú organizáciu,
   b) pomocou systémového triediaceho programu ich zotriedime,
   c) vytvoríme výstup za organizáciu,
   d) podľa okolností pokračujeme opäť od bodu b) (ak chceme za jednu organizáciu vytvárať viac výstupov), alebo od bodu a) (ak chceme prejsť k ďalšej organizácii).

   Toto riešenie umožňuje dostatočne pružnú prácu programov a neobmedzuje prakticky vôbec rozsah súborov, nevýhodou je veľká spotreba času, súvisiaca s tým, že na mnohonásobné triedenie súborov takého rozsahu, aký sa v praxi vyskytuje (niekoľko až niekoľko tisíc viet) nie je efektívne používať systémový triediaci program.
4. (Horí) Tým sa dostávame k riešeniu, o ktorom hovorí tento príspevok. Vychádza z nasledujúcich predpokladov:
   - časovú stratu pri použití systémového triediaceho programu podľa predchádzajúceho bodu spôsobuje viacnásobné čítanie jednej vety z disku resp. jej zápis na disk,
   - ak nahradíme toto čítanie a zápisy aj dosť rozsiahlymi operáciami vo vnútornej pamäti, ušetríme tedy čas.

   Postupujeme teda takto (obr. 3):
   a) načítame vety z organizácie na disk, a to do súboru (pri použití jazyka PL/I) typu REGIONAL (1), t.j. každá veta dostáva postupne poradové číslo, ktoré budeme volať jej diskovou adresou

b) súčasne s načítaním vety na disk zapíšeme do tabuľky kľúčov a tabuľky adries vo vnútornej pamäti
   - kľúč vety pre dané triedenie,
   - jej diskovú adresu
c) zotriedime tabuľku kľúčov podľa kľúčov viet so súčasným premiestnením diskových adries,
d) pomocou pretriedenej tabuľky adries môžeme teraz prečítať vety z disku v správnom poradí a vytvoriť výstup.

Trochu terminológie

Súbor je (z hľadiska triedenia)
- malý, ak môžeme všetky jeho vety zapísať do vnútornej pamäte a tam ich triediť,
- stredne veľký, ak môžeme použiť na jeho triedenie práve opísanú metódu,
- veľmi veľký, ak sme nútení triediť ho vo vonkajšej pamäti.

Rozsah malých a stredne veľkých súborov je obmedzený:
- rozsahom vnútornej pamäte: u malých súborov sa do nej musí zmestiť celý súbor, u stredne veľkých tabuľky kľúčov a adries,
- dovoleným rozsahom polí v programovacích jazykoch (napr. 32 K).

V zostávajúcej časti tohto príspevku sa budeme zaoberať dvoma témami:
- technickými detailmi triedenia,
- aplikáciami stredne veľkých súborov,
- metódami, ktoré umožňujú zvýšiť počet viet v stredne veľkých súboroch alebo zrýchliť ich triedenie.

Výber triediacej procedúry

Pre vnútorné triedenie by prichádzala do úvahy jedna z metód, pre ktoré počet porovnaní (a prípadne výmien) kľúčov rastie s počtom N kľúčov v tabuľke približne ako NlogN: SHELLSORT, QUICKSORT a jeho varianty, TREESORT a jeho varianty.

Vo všetkých prípadoch treba zabezpečiť súčasne výmeny v tabuľke kľúčov a adries.

Ak chceme príslušnú procedúru písať v niektorom vyššom jazyku, musíme brať do úvahy to, že prvky tabuľky kľúčov môžu mať v rôznych aplikáciách (aj v rámci toho istého programu) rôznu dĺžku a dokonca byť rôzneho typu (viď ďalej v odseku o kompresii kľúčov). Z toho dôvodu je vhodné triedenie zaraďovať na potrebné miesto ako otvorenú procedúru; dôležité je teda, aby algoritmus bol krátky a jednoduchý. Tomu najlepšie odpovedá SHELLSORT (obr. 4) v úprave podľa [1]. Treba tu poznamenať, že voľba triediacej procedúry sama o sebe nie je kritická. Po prechode od systémového diskového triedenia k internému sa trvanie typického programu skrátilo z 2 hodín na 45 minút. Naproti tomu u iného programu, ktorý už po úprave na interné triedenie trval približne 2,5 hod., neprinieslo nahradenie jednej varianty SHELLSORTU inou, ktorá bola pri testoch 2x rýchlejšia, žiadne viditeľné zlepšenie. Zdá sa, že po takejto úprave trávi program väčšinu času inde.

Pre voľbu SHELLSORTU hovorí aj to, že hoci QUICKSORT má lepšie vlastnosti v priemernom prípade, môže jeho správanie v najhoršom prípade byť v podstate tak zlé, ako u jednoduchých triediacich algoritmov rádu N^2. Pritom tieto prípady nastávajú pre často sa vyskytujúce takmer (alebo celkom) usporiadané tabuľky.

Aplikácie metódy

Metóda diskového triedenia stredne veľkých súborov sa dobre uplatní pri použití normovaného programovania. V našej situácii pri použití normovaného programovania

- na počiatku organizácie (blok GH [2]) sa vynuluje počítadlo tabuľky kľúčov,
- pri spracovaní prečítanej vety ju zapíšeme na disk a zapíšeme príslušné kľúče do tabuliek kľúčov,
- pri zmene organizácie (blok GZ [2]) vykonáme triedenie.

Je zrejmé, že takéto pretriedovanie je možné použiť nielen na vstupe, ale aj pri výstupe. Jedna z možných aplikácií sa dotýka tlače zostáv. Predpokladajme, že jedna procedúra vytvára (pre rôznych užívateľov) dve podobné (ale nie rovnaké) zostavy: základnú zostavu 011 a odvodenú zostavu 012, ktorá obsahuje

- iba niektoré riadky základnej zostavy (napr. sumy)
- poradie riadkov môže byť zmenené

(viď obr. 5a).

Potom môžeme postupovať takto (obr. 5b):

a) riadok zostavy zapíšeme na disk
b) pre každú zostavu, v ktorej sa riadok vyskytuje, zapíšeme do tabuľky kľúčov jeho kľúč (zostava + riadok) a adresu
c) po zotriedení dostaneme tabuľku zostáv, pomocou ktorej prečítame z disku príslušné riadky.

Poznámky:

1. Ako vidíme, samotný riadok sa vyskytuje na disku iba raz nezávisle od toho, v koľkých zostavách vystupuje.
2. Pre tlač zostáv je výhodné vedieť, aká časť tabuľky ktorej z nich patrí. To nám umožnia zistiť počítadlá, ktoré udávajú počet zápisu pre každú zostavu.
3. O niečo jednoduchší postup nám umožní vytvárať od danej zostavy vopred určený počet kópií.

Súčasné triedenie viacerých súborov

Podobnú techniku, ako pre generovanie viacerých zostáv, možno použiť aj v prípade, že chceme v jednom programe triediť niekoľko stredne veľkých súborov súčasne:

a) pre všetky súbory vytvoríme spoločnú tabuľku kľúčov, v ktorej sa každý kľúč skladá
   - z označenia súboru,
   - z kľúča v tomto súbore (upraveného na dĺžku najdlhších kľúčov)
b) pre každú spracovanú vetu zapíšeme do tejto tabuľky príslušný kľúč. Súčasne napočítavame do špeciálnych počítadiel počty viet jednotlivých súborov

c) po zápise poslednej vety tabuľku zotriedime.

Poznámka:

V prípade, že sa dĺžky kľúčov v súboroch veľmi líšia, vedie tento spôsob k nehospodárnemu využitiu vnútornej pamäte.

Úsporné využitie vnútornej pamäte

Aby sme mohli popísanou technikou triediť čo najrozsiahlejšie súbory, musíme sa usilovať čo najviac šetriť vnútornou pamäťou, obsadenou tabuľkami kľúčov a adries. K tomu existujú rôzne metódy; bez toho, že by sme si chceli robiť nárok na úplnosť, rozdelíme ich na dve triedy:
- lokálne - skracovanie jednotlivých kľúčov resp. adries, a
- globálne - zmena organizácie tabuliek.

Lokálne metódy

a) Zhusťovanie numerických kľúčov.
Ak sa kľúče skladajú iba z číslic a ich dĺžka nepresahuje 15 znakov, možno ich zapísať v zhustenom tvare a pri použití opäť previesť do zánového tvaru. Tento spôsob šetrí miesto, ale za cenu dodatkového času na konverziu.

b) Zhusťovanie adries.
Adresa (kľúč) pre súbor typu REGIONAL(1) je typu PICTURE '(8)9'. V prípade (v praxi najčastejšom), že vystačíme s 99999 vetami, môžeme deklarovať tabuľku adries ako FIXED(5), takže každá adresa zaberá 3 byty.

c) Kompresia s ohľadom na rozsah.
Uvažujeme opäť numerické kľúče. "Rozpočtová skladba" v našej vete má nasledujúcu štruktúru:

Kód	rozsah	Počet bitov
kapitola	0 - 51	6
oddiel	0 - 17	5
paragraf	0 - 51	6
článok	0 - 51	6
spolu		23

Takýto kľúč môžeme ľahko umiestniť ako jedno číslo typu BINARY FIXED (4 byty).
Ak pridávame ešte kód "položka" (rozsah 0 - 999, čiže 10 bitov), potom počet bitov dosiahne 33, čo prekročí rozsah čísla BINARY FIXED. Keďže také číslo nepresiahne 10 cifier, môžeme použiť pre jeho uloženie deklaráciu FIXED (11) (6 bytov) oproti 11 bytom, ktoré sú potrebné na uloženie znakovej konštanty.

d) Kompresia s použitím tabuliek.
Využime to, že kombinácií
kapitola + oddiel + paragraf + článok
je v skutočnosti iba okolo 400 a že všetky môžeme vopred udať (vo forme tabuľky). Podobne položiek a pseudopoložiek, ktoré používame, je okolo 500. Za týchto okolností môžeme každú zložku kľúča nahradiť jeho poradovým číslom podľa príslušnej tabuľky, čo nám umožní celý kľúč skrátiť na 16 bitov a pomocou deklarácie FIXED (5) "stlačiť" na 3 byty.
Takéto vyhľadávanie v tabuľke môžeme vykonávať
- vždy pri kódovaní kľúča (s použitím procedúry pre hľadanie polením intervalu v jazyku ASSEMBLER je spotreba času minimálna)
- "raz navždy" pri nahrávaní pohybovej vety (ako "vedľajší produkt" kontroly správnosti rozpočtovej skladby).

Globálne metódy

V tomto odstavci opíšeme (pre ilustráciu) iba jednu skutočne vyskúšanú úlohu. Využíva dva fakty:
- "čiastkové súbory", ktoré treba zotriediť, môžu byť v skutočnosti menšie. V rámci každej organizácie sú napr. vety zotriedené podľa syntetického účtu a pre určitý program sa toto triedenie nemení.
- pre prácu s už "zotriedeným" súborom potrebujeme vlastne iba tabuľku adries. Postupujeme preto takto:

- triedime vždy kľúče viet za jeden syntetický účet,
- zotriedené adresy ukladáme postupne do celkovej tabuľky adries. Tá môže byť podstatne dlhšia, ako tabuľka kľúčov.

Záver

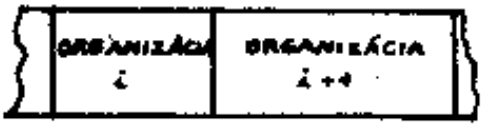
Predkladaný príspevok je iba stručným popisom konkrétnej aplikácie metódy triedenia stredne veľkých súborov. Ak sa niekomu podarí jej prenesením do iného kontextu vyriešiť svoje problémy, nebol napísaný zbytočne. Súčasne sa autor ospravedlňuje všetkým tým, ktorí museli prečítať celý príspevok a nakoniec zistiť, že ide o niečo, čo oni už dávno poznajú.

Literatúra

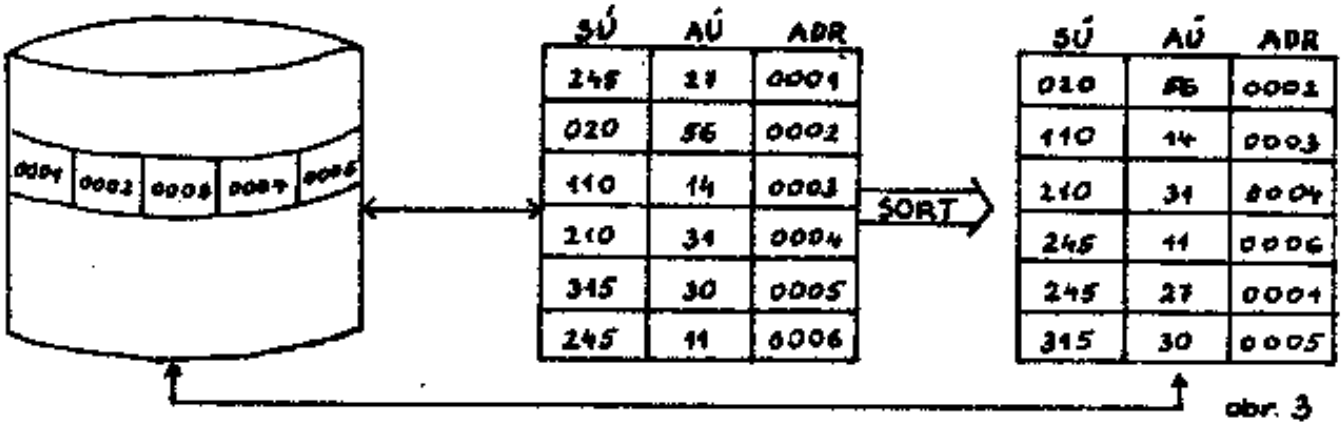
[1] D.E. Knuth: Iskusstvo programirovanija dľa EVM, zv.3: Sortirovka i poisk, Moskva 1978

[2] Normované programování MAA 6/1971

obr. 1



obr. 2

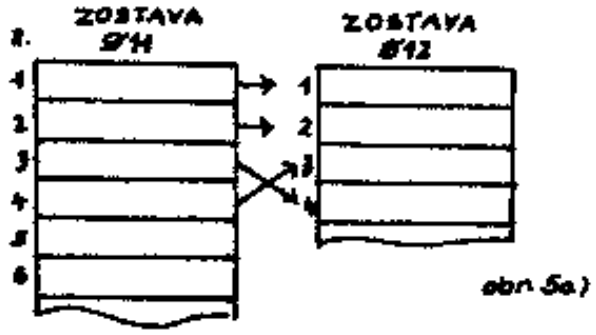


obr. 3

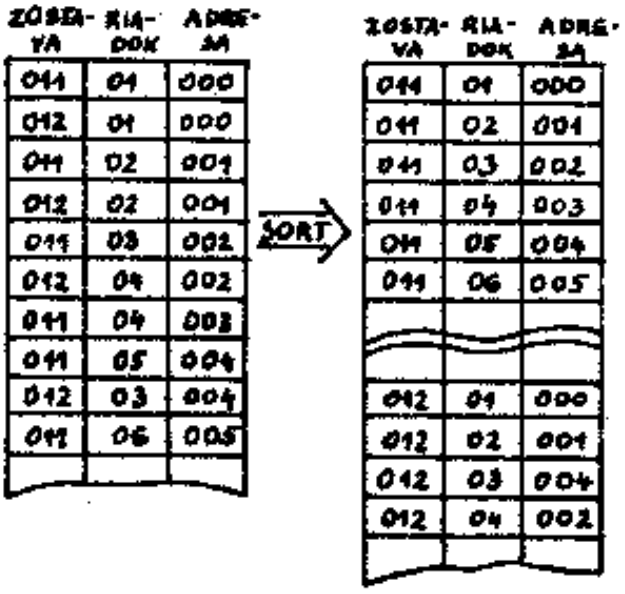
```
DECLARE
MH(9) INIT (1,4,13,40,121, 364, 1093,
            3280, 9841);
```

```
DO IT=2 TO 8;
IF MH(IT+1)>=N THEN GO TO SRTF;
END;
SRTF: IT=IT-1; M = MH(IT);
      DO J=1 TO N-M;
      KK = KLUC(J+M); KA=ADR(J+M);
      DO I=J BY -M TO 1;
      IF KLUC(I) < KK
         THEN GO TO SRTH;
      KLUC(I+M) = KLUC(I);
      ADR(I+M) = ADR(I);
     END;
 SRTH: KLUC(I+M) = KK;
       ADR(I+M) = KA;
     END;
IF M>1 THEN GOTO SRTF;
```

obr. 4



obr. 5a)



obr. 5b)