Razvoj softvera 2

Metodi pretraživanja tabela

Andrej Panić 1074/2014

Metod pretraživanja tabele

- je sema koja omogućava pretraživanje informacija u tabelama, da bi se izbeglo korišćenje logičkih naredbi (if ili case). Sve što se može uraditi sa logičkim naredbama može da se uradi i sa tabelama. Kako se logika komplikuje , to su tabele praktičnije za upotrebu.
- Da bismo klasifikovali karaktere u slova, znake interpunkcije i cifre, možemo koristiti komplikovanu logiku:

Primer klasifikacije karaktera

Java kod

```
if ( ( ( 'a' <= inputChar ) && ( inputChar <= 'z' ) ) ||
 ( ( 'A' <= inputChar ) && ( inputChar <= 'Z' ) ) ) {
charType = CharacterType.Letter;
}
else if ( ( inputChar == ' ' ) || ( inputChar == ',' ) ||
 ( inputChar == '.' ) || ( inputChar == '!' ) || ( inputChar == '(' ) ||
 ( inputChar == ')' ) || ( inputChar == ':' ) || ( inputChar == ';' ) ||
 ( inputChar == '?' ) || ( inputChar == '-' ) ) {
charType = CharacterType.Punctuation;
}
else if ( ( '0' <= inputChar ) && ( inputChar <= '9' ) ) {
charType = CharacterType.Digit;
}
```

Primer klasifikacije karaktera

- Ako koristimo pretraživačke tabele, možemo da sačuvamo tip svakog karaktera u niz, kome se pristupa po tipu karaktera.
- charType = charTypeTable[inputChar];
- Ovde se podrazumeva da je niz charTypeTable inicijalizovan ranije.

Problemi koji se javljaju

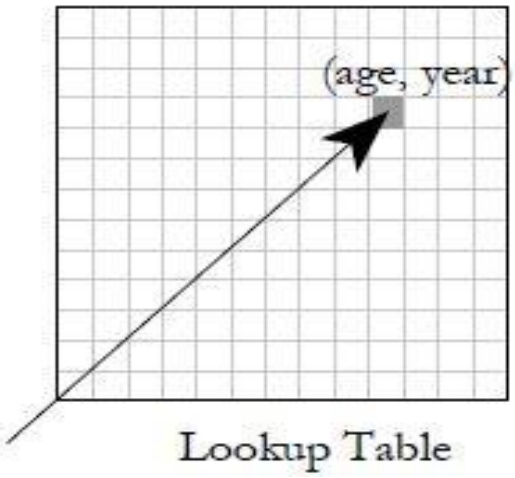
- Prvi problem je način na koji cemo tražiti vrednosti u tabeli. Mogu se koristiti odredjeni podaci da bi se tabeli direktno pristupilo. Ako podatke klasifikujemo po mesecima, mozemo koristiti niz sa indeksima od 1 do 12.
- Ako podatke moramo da klasifikujemo po broju socijalnog osiguranja, tada koristimo druge pristupe. Mogući su sledeći načini:
- Direktni pristup
- Indeksni pristup
- Stepenasti pristup

Problemi koji se javljaju

- Drugi problem koji se javlja je šta čuvati u tabeli. U nekim slučajevima rezultati pretraživanja su podaci. Tada se podaci mogu staviti u tabelu.
- Rezultat pretraživanja može biti i akcija. U tom slučaju čuvamo kod, koji opisuje akciju ili referencu na funkciju koja implementira tu akciju. Tabela tada postaje dosta komplikovanija.

Direktan pristup

Direktan pristup omogućava da uvek možete pristupiti elementu koji vas ineresuje



- Primer dani u mesecu: Odrediti broj dana u mesecu. Nespretan način je napisati veliku if naredbu

Dani u mesecu

```
If (month = 1) Then
   days= 31
ElseIf ( month = 2 ) Then
   days=28
ElseIf ( month = 3 ) Then
   days = 31
ElseIf ( month = 4 ) Then
   days = 30
ElseIf ( month = 5 ) Then
   days = 31
ElseIf ( month = 6 ) Then
   days = 30
ElseIf ( month = 7 ) Then
   days = 31
ElseIf ( month = 8 ) Then
   days = 31
…
End If
```

Direktan pristup

- Lakši i modifikovaniji način da se to uradi je da postavimo podatke u tabelu. U Visual Basic-u, prvo postavljamo tabelu
- Dim daysPerMonth() As Integer = _{ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }
- Sada umesto duge if naredbe, imamo samo days = daysPerMonth(month-1). Za prestupnu godinu imamo days = daysPerMonth(month-1, LeapYearIndex()) , gde LeapYearIndex () može imati vrednosti 0 ili 1
- Možemo koristiti month promenljivu da pretražimo neki podatak iz tabele.

Rata za osiguranje

Želimo da računamo ratu za medicinsko osiguranje, i imamo rate koje variraju u odnosu na godine, pol,bračno stanje, da li osoba pusi ili ne.

```
if ( gender == Gender.Female ) {
if ( maritalStatus == MaritalStatus.Single ) {
if ( smokingStatus == SmokingStatus.NonSmoking ) {
if ( age < 18 ) {
rate = 200.00;
}
else if ( age == 18 ) {
rate = 250.00;
}
else if ( age == 19 ) {
rate = 300.00;
}...
```

Rata za osiguranje

```
else if ( 65 < age ) {
rate = 450.00;
}
else {
if ( age < 18 ) {
rate = 250.00;
 }
else if ( age == 18 ) {
rate = 300.00;
 }
else if ( age == 19 ) {
rate = 350.00;
}...
else if ( 65 < age ) {
rate = 575.00;
}
}
else if ( maritalStatus == MaritalStatus.Married )...}
```

Rata za osiguranje

Možemo da stavimo rate u odvojene nizove za svaku od godina, ali je bolje rešenje da ih stavimo u nizove za svaki od faktora koje imamo.

```
Public Enum SmokingStatus
        SmokingStatus_First = 0
        SmokingStatus_Smoking = 0
        SmokingStatus_NonSmoking = 1
        SmokingStatus_Last = 1
End Enum
Public Enum Gender
        Gender_First = 0
        Gender_Male = 0
        Gender_Female = 1
        Gender_Last = 1
End Enum
Public Enum MaritalStatus
        MaritalStatus_First = 0
        MaritalStatus_Single = 0
        MaritalStatus_Married = 1
        MaritalStatus_Last = 1
End Enum
```

Rata za osiguranje

Const MAX_AGE As Integer = 125

Dim rateTable (SmokingStatus_Last, Gender_Last, MaritalStatus_Last, _MAX_AGE) As Double

Podaci se mogu smestiti u niz naredbom dodele, čitanjem sa diska, računanjem podataka. Kada smo to sredili, računamo ratu sa :

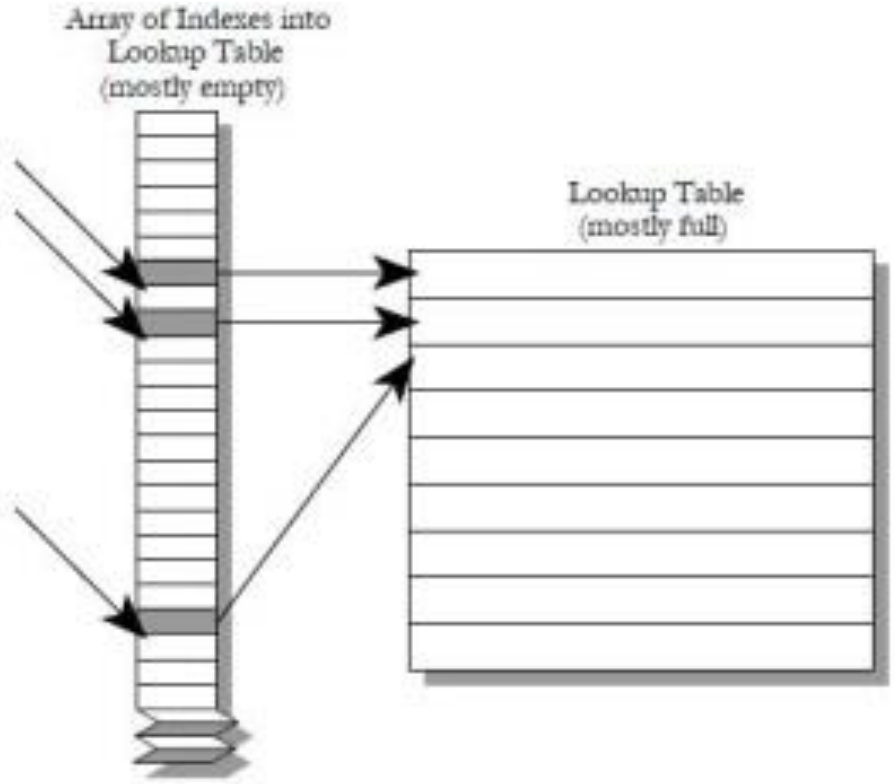
rate = rateTable(smokingStatus, gender, maritalStatus, age)

Pretraživačka tabela je lakša za čitanje i modifikovanje, zauzima manje prostora i brže se izvršava.

Indeksni pristup tabelama

- Kada koristite indekse, koristite primarne podatke da biste našli ključ indeksne tabele i zatim na osnovu vrednosti iz indeksne tabele, tražite glavni podatak koji vas interesuje.
- Pretpostavimo da vodite skladište i imate inventar od oko 100 stavki. Svaka stavka sadrži četvorocifreni serijski broj od 0000 do 9999. U ovom slučaju , ako želite da vam serijski broj bude ključ u tabeli koja opisuje neke aspekte svake stavke, setujemo indeksni niz sa 10,000 unosa (od 0 do 9999). Niz je prazan izuzev za 100 unosa koji odgovaraju serijskim brojevima za 100 stavki u skladištu. Na sledecoj slici se vidi da ovi unosi pokazuju na tabelu opisa stavki, koja ima daleko manje od 10,000 unosa.

Indeksni pristup tabelama



Tabelama pristupamo pomoću indeksa. Postoje dve prednosti ovakvog pristupa. Prvo, ako je bilo koji unos u glavnoj pretraživačkoj tabeli veliki, potrebno je mnogo manje prostora da se napravi indeksni niz, sa puno izgubljenog prostora nego što je potrebno da se napravi glavna pretraživačka tabela, sa puno izgubljenog prostora. Na primer, neka glavna tabela zauzima 100B-ova po unosu i neka indeksni niz zauzima 2B po unosu. Dalje, neka glavna tabela ima 100 unosa i da podaci korišćeni za pristup imaju 10,000 mogućih vrednosti.

Indeksni pristup tabelama

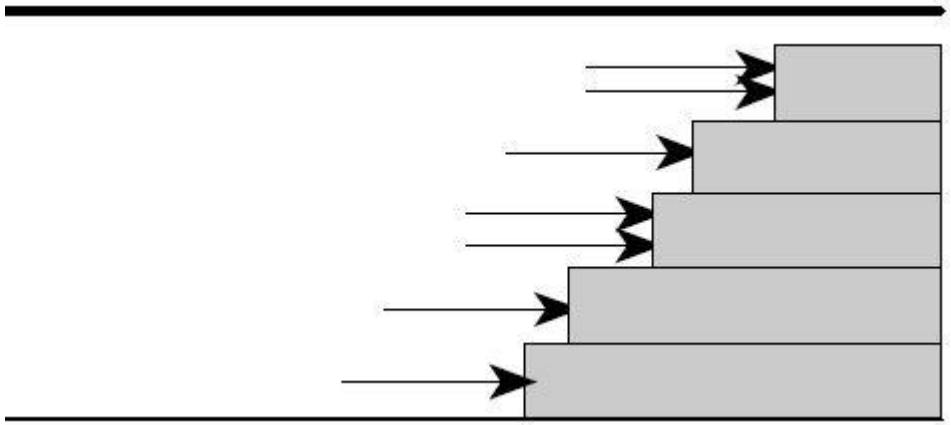
- U tom slučaju, izbor je između indeksa sa 10,000 unosa ili glavne tabele sa isto toliko unosa. Ako koristimo indeks, ukupna iskorišćena memorija je 20,000B. U suprotnom, ako se odreknemo strukture indeksa i izgubimo prostor u glavnoj tabeli, ukupna iskorišćena memorija je 1,000,000B.
- Druga prednost je ta da čak i da ne sačuvate prostor korišćenjem indeksa, ponekad je jeftinije da manipulišete unosima u indeksu nego u glavnoj tabeli. Na primer, ako imate tabelu sa imenima zaposlenih ,datumima zapošljavanja i platama, možete napraviti jedan indeks koji pristupa tabeli preko imena zaposlenog, drugi koji pristupa tabeli preko datuma zapošljavanja,i treći koji pristupa preko plate.

Indeksni pristup tabelama

- Konačna prednost šeme pristupa preko indeksa je generalno prednost održavanja glavne tabele. Podaci kodirani u tabelama se lakše održavaju nego podaci ugrađeni u kod. Kod za pristupanje indeksu može da se stavi u sopstvenu funkciju i ta funkcija da se poziva kada je potrebno dobiti ključ iz tabele za serijski broj.
- Kada je potrebno promeniti tabelu, može se promeniti šema pristupa indeksu ili da se izvrši promena na još jednu šemu pretraživačke tabele u celosti. Pristup šemi će biti lakši, ako se pristupi indeksu ne šire kroz program.

Stepenasti pristup

Ova metoda nije direktna kao indeksna struktura – ali ne troši toliko memorijskog prostora. Generalna ideja ,koja je prikazana na slici, je da su unosi u tabeli validni za raspon podataka.



Stepenasti pristup

- Na primer, ako pišemo program za ocene, granice unosa B mogu biti od 75 do 90 procenata. Na primer:

≥ 90.0%	A
< 90.0%	B
< 75.0%	C
< 65.0%	D
< 50.0%	F

Ovo su loše granice za pretraživačku tabelu , zato što se ne mogu koristiti jednostavni podaci- funkcija transformacije ključa u slova A do F. Indeksna šema bi bila čudna zato što su brojevi sa decimalnom tačkom.
U ovoj metodi, u tabelu postavljamo sve gornje granice , zatim pomoću petlje proveravamo rezultate poredeći sa tim gornjim granicama. Kada pronadjemo tačku u kojoj rezultat premašuje gornju granicu, znamo koja je ocena. Gornje granice treba izabrati da budu odgovarajuće.

Stepenasti pristup

```
' set up data for grading table
Dim rangeLimit() As Double = { 50.0, 65.0, 75.0, 90.0, 100.0 }
Dim grade() As String = { "F", "D", "C", "B", "A" }
maxGradeLevel = grade.Length – 1
...
' assign a grade to a student based on the student's score
gradeLevel = 0
studentGrade = "A"
While ( ( studentGrade = "A" ) and ( gradeLevel <
maxGradeLevel ) )
If ( studentScore < rangeLimit( gradeLevel ) ) Then
studentGrade = grade( gradeLevel )
 End If
gradeLevel = gradeLevel + 1
Wend
```

Prednost ovog pristupa je što radi dobro i sa neregularnim podacima. Primer sa ocenama je jednostavan, mada su ocene dodeljene neregularnim intervalima, brojevi su ‘okrugli’, završavaju sa 0 i sa 5.

Stepenasti pristup

Probability	Insurance Claim Amount
0.458747	$0.00
0.547651	$254.32
0.627764	$514.77
0.776883	$747.82
0.893211	$1,042.65
0.957665	$5,887.55
0.976544	$12,836.98
0.987889	$27,234.12
...	

Ovakvi brojevi ne mogu se dobro transformisati u tabelarne ključeve. Ovde je stepenasti pristup odgovor.
Ako se granice ocena moraju promeniti, program se lako može prilagoditi modifikovanjem unosa u RangeLimit nizu.
Bitno je pokriti slučaj na svakoj gornjoj granici ove metode. Pretražiti tako da stavke budu u bilo kojoj granici izuzev najveće, i da ostatak stavki upadne u najveću granicu. Ponekad ovo zahteva postavljanje veštačke vrednosti za najveću granicu.Voditi računa da se ne pravi greška sa < I <=.
Petlja mora da se završava odgovarajuće, sa vrednostima koje upadaju u gornje granice i da granice budu korektno postavljene.

Stepenasti pristup

- Koristite binarnu pretragu pre sekvencijalne pretrage

U primeru sa ocenama, petlja koja dodeljuje ocene, sekvencijalno prolazi kroz granice ocena. Ako bismo imali veću listu, sekvencijalna pretraga je spora. Ona se može zameniti kvazi-binarnom pretragom. Kod “kvazi” binarne pretrage, očekujemo da pronadjemo pravu kategoriju za vrednost. Algoritam mora korektno da odredi gde vrednost treba da upadne.

- Koristite indeksni pristup umesto stair-step tehnike

Brzina izvršavanja pretrage u stair-step metodi moze biti zabrinjavajuća. Može se dodati indeksna struktura koja će je ubrzati sa svojim direktnim pristupom. Alternativa nije dobra u svim slučajevima. Ako imate 100 diskretnih poena izraženih u procentima, cena memorije koju indeksni niz zauzima se neće isplatiti. U slučaju brojeva kao što su 0.458747 i 0.547651, ne može se setovati indeksna šema zbog ključeva.
“Bolje je težiti dobrom rešenju i izbeći katastrofu nego tragati za najboljim rešenjem” (Lampson 1984.)

Zaključak

- Tabele obezbedjuju alternativu komplikovanoj logici i nasledjenim strukturama.
- Jedno važno razmatranje o korišćenju tabela je način na koji im pristupamo. To može biti direktno, indeksno ili stepenasti pristup.
- Još jedna važna stvar je šta tačno staviti u tabelu.