Select ime id sum(cena) as sk.vr? From dobavitelji || Join || Where leto 2020 || Group by id || Having sk.vr > 500

Stik z vgnezdeno zanko po vrsticah: M + M*prm*N (M- zunanja relacija, N- notranja rel.)

1.1 **Stik z vgnezdeno zanko po blokih**: M+M*N Za vsako stran P1 preberi vsako stran P2 in izpiši spete pare n-teric • če imamo v pomnilniku dovolj prostora za N+2 strani, pri čemer je N manjša relacija: M+N • če ni dovolj prostora (recimo za B=102): M+N*(M/(B-2)) = 1000+500*1000/100=6000

1.2 **Stik z vnezdeno zanko z indeksom** (če imamo na enem od atributov stika že razpršilni indeks, na notranjo stran stika damo relacijo z indeksom) N + (M*prm*1,2) (prm – zapisi na stran)

1.3 **Stik z zlivanjem**: M and N (primer imamo B=100) ~**2* M (1+logB-1(|M|/B))** + 2*|N|* (1+logB-1(|N|/B)) + |M| +|N|= 2*1000*(1+1) + 2*500*(1+1) + 1000 + 500 = 7500 ~2

Stik z razpršilnim indeksom če imamo indeks na obeh id-jih, ki stikata tabeli • 3*(M+N), če B > √N (N je manjša relacija) • Če ne, imamo primer prekoračenja in vsako particijo manjše relacije, ki ne pride v pomnilnik, rekurzivno razdelimo po istem postopku.

1NO - ima izbran primarni ključ in nima večvrednostnih atributov (ima določen ključ)

2NO - nima parcialnih odvisnosti, vsi neključni atributi so odvisni ok ključa(ima lahko 1 ali več attr), vsi atributi so odvisni od celotnega ključa

3NO - nima tranzitivnih odvisnosti. - odvisnost med atributi, ki niso del ključa

3.5NO – vse entitete morajo biti ključ(vse podčrtane)Vsak dejavnik mora biti kandidatni ključ. Glej 2NO, tu mora biti D odvisen od A,B in C !

4NO - Razmerje je v 4NF, če nima več odvisnosti.

Armstrongovi aksiomi

A1: refleksivnost $Y \subseteq X \Rightarrow X \rightarrow Y$ A2: razširitev $\{X \rightarrow Y\} \vDash XZ \rightarrow YZ$ A3: tranzitivnost $\{X \rightarrow Y, Y \rightarrow Z\} \vDash X \rightarrow Z$

I1: dekompozicija: $\{X \rightarrow YZ\} \vDash X \rightarrow Y$

A1 : $YZ \rightarrow Y$

A3 : $\{X \rightarrow YZ, YZ \rightarrow Y\} \vDash X \rightarrow Y$

• I2: združitev: $\{X \rightarrow Y, X \rightarrow Z\} \vDash X \rightarrow YZ$

A2 : $\{X \rightarrow Y\} \vDash X \rightarrow YX$

A2 : $\{X \rightarrow Z\} \vDash YX \rightarrow YZ$

A3 : $\{X \rightarrow YX, YX \rightarrow YZ\} \vDash X \rightarrow YZ$

• I3: psevdotranzitivnost: $\{X \rightarrow Y, WY \rightarrow Z\} \vDash WX \rightarrow Z$

A2 : $\{X \rightarrow Y\} \vDash WX \rightarrow WY$

A3 : $\{WX \rightarrow WY, WY \rightarrow Z\} \vDash WX \rightarrow Z$

VELIKOST INDEKSA potrbujemo PID in RID , katerima pristejemo velikost atributa(velikost vrstice/ st attributov). Kar dobimo je velikost indeksnega(v drevesu) in podatkovnega(iz drevesa v liste) vpisa. Nato izračunamo, koliko le the lahko damo v eno stran (stran/pod.vpis). Nato stevilo podatkovnih strani / podatkovnimi vpisi na stran, potem pa za vsak nivo visje uporabimo indeksne vpise.

PID (identifikator strani) RID (identifikator zapisa) PVS (1 str na disku / (PID + velikost u bajtih)) IVS (1 str na disku / (rid + velikost u bajtih)) za računat rabi zapisi/PVS prvič ostale zapisi/IVS.

STATIČNI RAZŠIRLJIV RAZPRŠILNI INDEKS

Ko se zapolni skupek, se mu dvigne lokalna globina(ustvarimo nov skupek in oba imata za 1 povecano lokalno globino in potem vse elemente prerazporedimo)(povečamo st bitov, ki jih gledamo) in ko lokalna globina postane večja od globalne, povečamo direktorij za 2x(direktorij vel 8 ima 3 bite, povecamo na 16 in imamo 4 bite na voljo). El vstavimo tako, da ga pretvorimo v binarno in ga vstavimo v stran na katero kaze to polje v direktoriju. 2. nacin: stevilo mod (velikost direktorija)

npr 10 mod 8 = 2.

LINEARNI RAZPRŠLJIV INDEKS

x mod (2^level * N) N= stevilo glavnih strani, level = 0, next = 0

ko se ustvari prelivna stran, ali pa je dosezen recimo pogoj polnosti, razcepimo stran, na katero kaze pointer next. Elemente te strani porazporedimo z x mod (2^level+1 * N) v različne strani. Next++, Ko razdelimo vseh n strani (next = N) → next = 0, LEVEL++ in N= N*2

TRANSAKCIJE

X(a) popolni zaklep, S(a) zaklep za branje.

Ti hoče zaklep, ki ga drži Tj

POČAKAJ -UMRI

če ima Ti višjo prioriteto, kot Tj, počaka na Tj, čene pa se prekine

RANI-ČAKAJ

če ima Ti, višjo prioriteto, se Tj prekine, če ima pa nižjo pa počaka

Razporedi so kofliktno ekvivalentni , če vsebujeta enake akcije na istih transakcijah in so pari konfliktnih akcij urejeni na enak način. Razpored je uredljiv po konfliktih, če je konf. Ekviv. Nekem zaporednem razporedu. Čakalni graf

podatkovna neodvisnost Fizična in 2. Logična neodvisnost

Nivoji abstrakcije: 1.Pogledi:Kako uporabnik vidi podatke(npr. direktor vidi podatke o svoji firmi)

2.Konceptualna shema: imamo atribut(ime) in tip atributa(string), definira logično strukturo.

3.Fizična shema: podatke vidimo tako kot so predstavljeni, opisuje uporabljene neurejene datoteke in indeksne datoteke.

Atomarna(se izvedejo vse ali nobena) sekvenca akcij za branje in pisanje. Vsaka transakcija, ki se izvrši celotno mora pustiti PB v konsistentnem stanju(omejitve nad podatki niso kršene.

Struktura SUPB je diskovni proctor vmesni proctor datoteke in metode dostopa relacijske op. optimizator

Omejitev integritete je osnovno pravilo, da za vsak atribut določimo lastnosti, ki se preverjajo ob spreminjanju relacij. Manj možnosti za napake pri vnosu. Pk tk Sk

PRIMARY key: Množica atributov, ki enolično določa vsako vrstico relacijo. Superključ-če primarnemu ključu dodamo atribut{sid, pid}.

TUJI KLUČ: Definirajo strukturo relacijo. Iz neke tabele pokažemo na primarni ključ druge relacije.”log k”

REFERENČNA integriteta je lastnost podatkov, ki navaja, da so vsi njeni sklici veljavni. V kontekstu relacijskih baz podatkov zahteva, da mora referenčna vrednost obstajati, če se vrednost enega atributa (stolpca) relacije (tabele) sklicuje na vrednost drugega atributa (bodisi v isti ali drugi relaciji). [1]

PRAVILA: ● Za vsak entitetni tip kreiramo eno relacijsko shemo. Ime relacije naj bo ime entitetnega tipa (priporočilo). ● Atributi relacije so atributi entitetnega tipa. ● Opcijske atribute prevedemo v atribute, v katerih dovolimo vrednosti 'NULL' oz. neobvezna polja.● Ključ entitetnega tipa postane primarni ključ pripadajoče relacije. ● Tuji ključi entitetnega tipa postanejo tuji ključi relacije.

B+ ind. Najbolj razširjen indeks. Uravnoteženo drevo, indeksne strani so 50% napolnjene. Iskanje se začne v korenu: primerjava ključev vodi do lista s podatkovnimi vpisi.

1.Stik z vgnezdeno zanko z indeksom: Preberemo vse n terke relacije R, za vsako n terko mormo dostopat preko indeksa do relac ije S. 1. v primeru da uporabimo razpršilni indeks v povprečju preberemo 1-2 strani. 2 v primeru uporabe B+ drevesa uporabimo 2-4 strani. Preberemo še pod. Zapise (v primeru povezanega - 1 stran, nepovezanega - 1stran + vse n terke).

2.Stik z vgnezdeno zanko po blokih: za vsako stran R2 preberemo vsako stran R1 in izpišemo pare n-teric. Cena: Vse zun. Strani + št. Zun blokov * vse notranji bloki. Št zun. Blokov = št vseh stranih zun.tabele deljimo z (B-2(DOLŽINA VRSTE)) ozr. strani blokov.

Dekompozicija pomeni zamenjavo R z dvema ali več relacijama, kjer velja vsaj ena odvisnost. Relacijo R razgradimo v dve novi relaciji in če velja R1 R2 = R., Kar pomeni da mora biti vsak atribut R v R1 ali v R2.

Funkcionalna odvisnost (FD) določa razmerje enega atributa do drugega atributa v sistemu za upravljanje baz podatkov (DBMS). Funkcionalna odvisnost pomaga ohranjati kakovost podatkov v bazi podatkov. Funkcionalno odvisnost X od Y predstavlja X → Y. funk odvis. igra ključno vlogo pri iskanju razlike med dobrim in slabim oblikovanjem baze podatkov.

1.**Stik z vgnezdeno** zanko z indeksom: Preberemo vse n terke relacije R, za vsako n terko mormo dostopat preko indeksa do relacije S. 1. v primeru da uporabimo razpršilni indeks v povprečju preberemo 1-2 strani. 2 v primeru uporabe B+ drevesa uporabimo 2-4 strani. Preberemo še pod. Zapise (v primeru povezanega - 1 stran, nepovezanega - 1stran + vse n terke).

2.**Stik z vgnezdeno** zanko po blokih: za vsako stran R2 preberemo vsako stran R1 in izpišemo pare n-teric. Cena: Vse zun. Strani + št. Zun blokov * vse notranji bloki. Št zun. Blokov = št vseh stranih zun.tabele deljimo z (B-2(DOLŽINA VRSTE)) ozr. strani blokov.

1.Sortiramo obe tabeli. **Stik zlivanje p**arov, ki se ujemajo s ključi shranimo kot rezultat. 2.(branje vsake posebaj) Najprej preberemo R dokler je n terka >= S nterki. Potem preberemo vse ključe dokler S n terka >= R terki. Nato so ključi R terk = S terkam. Nadaljujemo da najdemo vse enake ključe iz n terk. Na koncu je potrebno prebrat še obe tabeli.

stik z razpršilnim indeksom - Uporabimo razpršilno funkcijo h, tako da razpršimo obe relaciji na B-1 particij. Preberemo eno particijo in vse zapise razpršimo z raz. Fun. H2. Beremo pripadajoče particije iz druge relacije, ki se bojo joinale in jih ponovno razpršimo z raz. Funk. H2. Te vsebujejo n terke ki jih joinamo skupaj.

Relacija je v tretji normalni obliki 3NO, če: • je v drugi normalni obliki 2NO in • nima tranzitivnih odvisnosti. Tranzitivna odvisnost je funkcionalna odvisnost med atributi, ki niso del ključa. Tranzitivne odvisnosti odpravimo tako, da relacijo razbijemo na več novih relacij. Relacija je avtomatsko v 3NO, če je v 2NO in je njen ključ sestavljen iz vseh ali vseh razen enega atributa sheme.

Dekompozicija pomeni zamenjavo R z dvema ali več relacijama, kjer velja vsaj ena odvisnost. Relacijo R razgradimo v dve novi relaciji in če velja R1 R2 = R., Kar pomeni da mora biti vsak atribut R v R1 ali v R2.

ohranjanje funkcijske odvisnosti - Definirana na osnovi projekcije množice odvisnosti: preverjanje ob vstavljanju in spreminjanju. Unija projekcij mora pokrivati vse odvisnosti. Če razstavimo relacijo R v relaciji R1 in R2, morajo biti vse odvisnosti R del R1 ali R2 ali pa jih je mogoče izpeljati iz kombinacije FD R1 in R2.

BCNO - Vsaka relacija razcepi relacijo in algoritem zaključi brezizgubni stil(ker velja vsaj ena odvisnost) koraki algoritma tvorijo drevo(rekonstrukcija).

3NF - Algoritem prej zaključi kot pri BCNO. Ta pristop ne zagotavlja ohranitev odvisnosti. Enostavna dekompozicija z brezigubnim stikom in z ohranitvijo odvisnosti.

horizontalna dekompozicija - Včasih lahko relacijo zamenjate z zbirko relacij, ki so selekcije. -Vsako novo razmerje ima isto shemo kot original, vendar podmnožica vrstic. -Skupaj novi odnosi vsebujejo vse vrstice originala. Nove relacije so navadno ločene(disjoint).

smiselna denormalizacija - Kadarkoli normalizirate podatke, morate vedno uskladiti konkurenčne cilje zmanjšanja odvečnosti podatkov z dodatnim delom, ki bo ustvarjeno, ko boste morali izvleči podatke iz baze. Elementi, kot so mesto, država in poštna številka, se večinoma prenašajo odvečno. Razmislite o primeru poskusa dodelitve mesta stranki, vendar mesto ni opredeljeno v mestni tabeli. Najprej morate vzdrževati mestno tabelo. Zdaj bi to, kar bi trajalo samo eno operacijo, trajalo dve. Zmanjšanje zahtevnosti poizvedb in izboljšanje uspešnosti poizvedb sta dobra razloga za denormalizacijo. Ker kršite pravilo, bi to morala biti izjema, ne pa norma

indekse ISAM- Drevesna struktura, sčasoma drevo postane neuravnoteženo. Statičen, sestavljen iz indeksnih zapisov-notr vozlišča in listi-podatkovnii vpis. Za izboljšanje na začetku, ko gradimo pustimo prazen prostor v straneh listov. Vsako vozlišče vsebuje dva v pisa, ni kazalcev, razen pri prelivnih straneh

B + drevesni indeks. - Najbolj razširjen indeks. Uravnoteženo drevo, indeksne strani so 50% napolnjene. Iskanje se začne v korenu: primerjava ključev vodi do lista s podatkovnimi vpisi. Listi vsebujejo podatkovne vpise(urejeni, prev, next), notra nje strani imajo indeksne vpise ozr. podatkovne zapise