# Fizičko projektovanje baza podataka

Ivana Tanasijevic, ivana@matf.bg.ac.rs
Matematički fakultet, Beograd

# Radno opterećenje

Opis radnog opterećenja treba da obuhvati sledeće:

1. Listu upita i njihove učestalosti u odnosu na sve upite i ažuriranja
2. Listu ažuriranja i njihove učestalosti
3. Cilj performansi za svaki tip upita ili ažuriranja.

Za svaki upit se mora odrediti:

- Kojim se relacijama pristupa
- Koji se atributi traže u SELECT iskazu
- Nad kojim atributima se vrši spajanje ili selekcija u WHERE iskazu i koliko su ti uslovi selektivni.

Za svako ažuriranje se mora odrediti:

- Nad kojim atributima se vrši spajanje ili selekcija u WHERE iskazu i koliko su ti uslovi selektivni
- Tip ažuriranja (INSERT, DELETE, UPDATE) i relacija koja se ažurira
- Za UPDATE komandu, polje koje treba ažurirati.

# Indeksi

Kada je fajl sa podacima organizovan tako da redosled zapisa odgovara redosledu zapisa u indeksu, tada se za indeks kaže da je klasterovan. Sa druge strane, ukoliko je redosled zapisa u fajlu proizvoljan i ne postoji razuman način da se sličan rakav redosled napravi i u indeksu, tada je indeks neklasterovan.

Fajl sa podacima može biti klasterovan samo po jednom ključu pretrage, odnosno može imati jedan klasterovani indeks, dok može imati i više neklasterovanih indeksa.

Za indeks se kaže da je gust ukoliko sadrži najmanje jedan zapis za svaku ključnu vrednost pretrage koja se pojavljuje u zapisu indeksnog fajla. Indeks je redak ako sadrži po jedan zapis za svaku stranicu fajla sa podacima. Ne može se napraviti redak indeks koji nije klasterovan, stoga može postojati samo jedan redak indeks.

# Odluke prilikom projektovanja

Važne odluke koje treba napraviti prilikom fizičkog projektovanja uključuju sledeće:

1. Koje indekse napraviti:

- Koje relacije indeksirati i koje polje ili koja polja izabrati za ključ pretrage indeksa
- Za svaki indeks odrediti da li treba biti klasterovan ili ne, da li treba biti redak ili gust.

2. Da li treba praviti sledeće promene u konceptualnoj shemi u cilju poboljšanja performansi:

- Alternativne normalizovane sheme
- Denormalizacija
- Vertikalno particionisanje
- Pogledi.

3. Da li treba preformulisati upite koji se često izvršavaju tako da budu efikasniji?

# Kreiranje indeksa

Uputstva za kreiranje indeksa bi bila sedeća:

**Uputstvo 1 (da li napraviti indeks)**: Ne treba praviti indeks ukoliko on neće doprineti ubrzanju izvršavanja nekog upita, uključujući i one koji su deo ažuriranja. Kada god je moguće, izabrati indeks koji poboljšava više od jedan upit.

**Uputstvo 2 (izabir ključa pretrage)**: Atributi koji se pominju u WHERE iskazu su kandidati za indeksiranje.

- Izdvajanje po tačnoj vrednosti sugeriše da treba razmotriti indeks nad izabranim atributima i to heš indeks.

- Intervalno izdvajanje sugeriše da treba razmotriti B+ stablo nad izdvojenim atributima.

**Uputstvo 3 (ključ pretrage od više atributa)**: Treba ga razmotriti u sledećim situacijama:

- WHERE iskaz sadrži uslov nad više od jednog atributa.

- Ako omogućava index-only strategije (tj. tamo gde se pristup relacijama može izbeći) u važnim upitima. Ovo može voditi ka atributima koji bi bili deo ključa iako se ne pominju u WHERE iskazu. Pri kreiranju indeksa nad ključevima pretrage sa više atributa, ukoliko se prihvataju i intervalni upiti, treba biti pažljiv oko redosleda atributa u ključu pretrage.

# Kreiranje indeksa

**Uputstvo 4 (da li klasterovati)**: Najviše jedan indeks nad datom relacijom može biti klasterovan, što veoma povećava performanse. Stoga treba pažljivo odabrati klasterovan indeks.

- Upiti intervala mogu veoma mnogo dobiti klasterovanim indeksom. Ukoliko se upotrebljava nekoliko intervalnih upita nad relacijom, koji uključuju drugačije skupove atributa, pri odabiru koji indeks treba biti klasterovan, treba razmotriti selektivnost upita i njihove relativne učestalosti u radnom opterećenju.

- Ukoliko indeks omogućava index-only strategiju za upit koji teba da ubrza, tada nije potrebno da bude klasterovan. Klasterovanje ima uticaja samo kod dobavljanja podataka.

**Uputstvo 5 (heš indeks naspram stabla)**: Uglavnom se preporučuje B+ stablo, zato što je dobro i za upite intervala i za upite sa tačnom vrednosti. Heš indeks je bolji u sledećim situacijama:

- Ovaj indeks bi trebalo da se koristi kod ugnježdene petlje spajanja, indeksirane relacije unutrašnjih relacija, gde ključ pretrage uključuje kolone po kojima se vrši spajanje. Razlog tome je taj što se izdvanje po jednakosti vrši za svaku n-torku u spoljnoj petlji.

- Kada postoje veoma važni upiti jednakosti, a pri tome ne postoje upiti intervala, uključujući atribute koji su ključ pretrage.

**Uputstvo 6 (balansiranje cene ažuriranja indeksa)**: Nakon sastavljanja liste željenih indeksa treba razmotriti njihov uticaj na ažuriranja u radnom opterećenju.

- Ukoliko održavanje indeksa usporava operaciju ažuriranja, razmotriti njegovo odbacivanje.

- Treba imati na umu da dodavanje indeksa može ubrzati i data ažuriranja.

# Uslov po jednakosti

```
SELECT E.ename, D.mgr
FROM   Employees E, Departments D
WHERE  D.dname='Toy' AND E.dno=D.dno
```

Treba napraviti heš indekse nad uključenim atributima

D.dname='Toy' AND E.dno=D.dno AND E.age=25.

Pored dname, indeksira se dno ili age, ali ne i oba.

# Intervalni uslov

```
SELECT E.ename, D.dname
FROM   Employees E, Departments D
WHERE  E.sal BETWEEN 10000 AND 20000
       AND E.hobby='Stamps' AND E.dno=D.dno
```

E treba da bude relacija u spoljašnjoj petlji, pošto postoje dva izdvajanja. Pravi se jedan indeks za E, B+ nad sal ili heš nad hobby. Nad D.dno se pravi heš indeks pošto je u unutrašnjoj petlji.

Ukoliko konstante nisu poznate, prave se oba indeksa i prepušta se optimizatoru da izabere.

# Klasterovani indeksi, grupisanje

```
SELECT E.dno
FROM   Employees E
WHERE  E.age > 40
```

Klasterovan indeks B+ stablo nad age. Posebno je loše ako indeks nije klasterovan.

```
SELECT E.dno, COUNT(*)
FROM   Employees E
WHERE  E.age > 10
GROUP BY E.dno
```

Klasterovan indeks B+ stablo nad age. Ukoliko uslov nad age nije mnogo selektivan, onda je bolje napraviti klasterovani indeks (B+ ili heš) nad dno.

# Klasterovani indeksi, više zapisa

```
SELECT    E.dno
FROM      Employees E
WHERE     E.hobby='Stamps'
```

Klasterovani indeks nad Stamps, ukoliko ima mnogo ljudi koji se bave markicama.

E.hobby='Stamps' and E.eid=552

Nije potrebno klasterovanje.

# Klasterovani indeks, unutrašnja petlja

```
SELECT E.ename, D.mgr
FROM   Employees E, Departments D
WHERE  D.dname='Toy' AND E.dno=D.dno
```

Ukoliko je broj torki koje zadovoljavaju uslov dname mali, tada je bolje napraviti neklasterovani indeks nad dname. Klasterovani indeks u unutrašnjoj petlji je veći prioritet od klasterovanog indeksa u spoljašnjoj petlji.

```
SELECT E.ename, D.mgr
FROM   Employees E, Departments D
WHERE  E.hobby='Stamps' AND E.dno=D.dno
```

Klasterovani indeks B+ stablo nad dno relacije Departments

# Ko-klasterovanje

```
SELECT    P.pid, A.componentid
FROM      Parts P, Assembly A
WHERE     P.pid = A.partid AND P.supplierid = ‘Acme’
```

```
SELECT P.pid, A.componentid
FROM      Parts P, Assembly A
WHERE     P.pid = A.partid AND P.cost=10
```

Pretpostavimo da mnogo delova ima cenu 10. Tada bismo napravili indeks nad cost i izdvojili delove koji zadovoljavaju uslov. Ko-klasterovanjem izbegavamo pravljenje indeksa nad poljem pid.

Sumarizovano ko-klasterovanje bi bilo:

- Može da ubrza spajanje, posebno spajanje strani ključ-ključ koje odgovara relacijama koje imaju kardinalnost 1:N
- Sekvencijalna pretraga bilo koje od relacija postaje sporija, ažuriranje je sporije

# Indeks sa više atributa

```
SELECT E.eid
FROM   Employees E
WHERE  E.age BETWEEN 20 AND 30
       AND E.sal BETWEEN 3000 AND 5000
```

Kompozitni indeks <age, sal> će pomoći ukoliko je uslov selektivan. Heš indeks neće pomoći, stoga je B+ stablo preporučljivo. Takođe, klasterovani indeks ima prednost nad neklasterovanim. Ukoliko su oba uslova podjednako selektivna nije važan redosled <age, sal> ili <sal, age>.

```
SELECT E.eid
FROM   Employees E
WHERE  E.age = 25
       AND E.sal BETWEEN 3000 AND 5000
```

Kompozitni klasterovani indeks B+ stablo <age, sal>

# Gusti indeksi, index-only zahtev

```
SELECT D.mgr
FROM   Departments D, Employees E
WHERE  D.dno=E.dno
```

Gust neklasterovani indeks nad poljem dno relacije Employees. Takođe primetiti da nije bitno da li je indeks klasterovan, pošto se ne dobavljaju torke.

```
SELECT D.mgr, E.eid
FROM   Departments D, Employees E
WHERE  D.dno=E.dno
```

Gusto B+ stablo nad <dno, eid>. Tada možemo da izračunamo upit koristeći umetnute petlje spajanja nad indeksima i to nad Departments kao spoljašoj relaciji i index-only pretragu kao unutrašnjoj relaciji.

# Gusti indeksi, index-only zahtev

```
SELECT E.dno, COUNT(*)
FROM   Employees E
GROUP BY E.dno
```

Gusti indeks, heš ili B+ stablo, možemo odgovoriti na upit samo pretragom indeksa i prebrojavanjem broja zapisa u samom indeksu

```
SELECT E.dno, COUNT(*)
FROM   Employees E
WHERE  E.sal=10,000
GROUP BY E.dno
```

Index-only plan ukoliko imamo kompozitno B+ stablo nad <sal, dno> ili <dno, sal>. Ukoliko koristimo prvi indeks, on se ne može koristiti ako se izmeni uslov sal > 10,000. Drugi indeks se može koristiti i u tom slučaju, s tim što je manje efikasan jer se moraju pregledati sve torke.

# Gusti indeksi, index-only zahtev

```
SELECT E.dno, MIN(E.sal)
FROM   Employees E
GROUP BY E.dno
```

Gusto kompozitno B+ stablo nad <dno, sal>.

```
SELECT AVG (E.sal)
FROM   Employees E
WHERE  E.age = 25
       AND E.sal BETWEEN 3000 AND 5000
```

Gusto kompozitno B+ stablo nad <age, sal> omogućava index-only plan.

# Podešavanje baze

Nakon početnog projektovanja baze podataka, sama upotreba može biti izvor veoma vrednih informacija o tome šta treba izmeniti u početnom dizajnu.

Neki od uslova u početnom radnom opterećenju se mogu pokazati kao netačni, dok se drugi mogu pokazati ispravnim.

Takođe, veličina podataka se ne mora poklapati sa onom koja je predviđena na početku.

Neprekidno podešavanje baze podataka je važno sa aspekta performansi.

Ovde ćemo pričati o podešavanju indeksa, podešavanju konceptualne sheme i podešavanju upita.

# Podešavanje indeksa

Početni izbor indeksa se može poboljšati iz više razloga.

Jedan od razloga je da je posmatrano radno opterećenje drugačiji od onog koji smo pretpostavljali.

Takodje, mogu se identifikovati neki drugi upiti kao važni, a koje nismo uvrstili u početnom dizajnu.

Takođe, može se ispostaviti da optimizator ne pravi odluke onako kako smo pretpostavljali.

# Podešavanje konceptualne sheme

- Možda je pogodnija 3NF umesto BCNF

- Ukoliko postoje dva načina da se shema dekomponuje treba izabrati onaj koji je bolji za konkretno radno opterećenje.

- Ponekad je dobro dalje dekomponovati relaciju koja je već u BCNF

- Ukoliko to poboljšava performanse, treba razmisliti o denormalizaciji, odnosno zameni više relacija koje su dobijene normalizacijom od jedne veće, tom većom relacijom.

- Razmatranje normalizacije se odnosi na dekompoziciju, što predstavlja vertikalno particionisanje. Druga tehnika koju treba razmotriti je horizontalna dekompozicija, kojom bi se dobile dve relacije sa istim shemama, ali sa na primer različitim ograničenjima ili indeksima.

# Podešavanje konceptualne sheme

Contracts(cid: integer, supplierid: integer, projectid: integer,

deptid: integer, partid: integer, qty: integer, value: real)

Departments(did: integer, budget: real, annualreport: varchar)

Parts(pid: integer, cost: integer)

Projects(jid: integer, mgr: char(20))

Suppliers(sid: integer, address: char(50))

F = {JP->C, SD->P }, gde je Contracts CSJDPQV

# Postavljanje slabije normalne forme

Ova relacija nije u BCNF, ali jeste u 3NF.

Da li treba da raščlanimo relaciju Contracts?

Kandidat ključa su C i JP. Relacija se može zameniti relacijama CJP, SDP, CSJDQV.

Ukoliko je jedan od važnih upita naći broj kopija Q dela P naručenog u ugovoru C, tada ćemo imati spajanje razdvojenih relacija, što je lošije nego da smo imali samo jednu relaciju.

# Denormalizacija

Možemo ići i dalje, tako da dozvolimo neka ponavljanja.

Neka je upit proveriti da je vrednost ugovora manja od budžeta odeljenja koje je obuhvaćeno ugovorom.

Možda možemo odlučiti da dodamo polje budžet u Contracts. Relacija tada neće biti u 3NF.

# Izbori pri dekompoziciji

- Možemo ostaviti Contracts takvu kakva jeste i prihvatiti postojeće ponavljanje

- Možemo je dovesti u BCNF sa očuvanjem svih zavisnosti ili bez nekih zavisnosti, koje na primer možemo modelovati uslovom ograničenja, koji je pak skup.

# Vertikalna dekompozicija

Pretpostavimo sa smo odlučili da raščlanimo Contracts na SDP i CSJDQV koje su u BCNF. Pretpostavimo da su sledeći upiti česti:

- Naći ugovore dobavljača S
- Naći ugovore odeljenja D.

Relacije tada možemo raščlaniti na CS, CD, CJQV.

# Horizontalna dekompozicija

Neka postoje različita pravila kod obrade ugovora čija je vrednost veća od 10,000 i onih čija je vrednost manja.

Jedno rešenje je da napravimo B+ stablo kojim čeo izdvajati interval.

Drugo rešenje je da napravimo dve relacije LargeContracts i SmallContracts.

Ovo je pogotovo primamljivo ako bi drugi upiti mogli da dobiju od uvođenja klasterovanog indeksa.

# Podešavanje upita i pogleda

```
SELECT E.dno
FROM   Employees E
WHERE  E.hobby='Stamps' OR E.age=10
```

Ukoliko imamo indekse nad hobby i nad age, optimizator to možda neće prepoznati i vršiće sekvencijalnu pretragu. Treba razmisliti o preformulacije upita tako da predstavlja uniju dva upita.

Takođe treba izbegavati skupe operacije, kao što je DISTINCT kada god je to moguće.

Ponekad se upit sa GROUP BY i HEAVING izrazima može preformulisati tako da ih ne koristi.

# Podešavanje upita i pogleda

Često se komplikovani upiti pišu u koracima, tako da se koriste privremene relacije. Kada je moguće, to treba izbegavati. Na primer, sledeći upit:

```
SELECT *
INTO      Temp
FROM      Employees E, Departments D
WHERE  E.dno=D.dno AND D.mgrname='Robinson'

SELECT T.dno, AVG (T.sal)
FROM      Temp T
GROUP BY T.dno
```

# Podešavanje upita i pogleda

Upit se može preformulisati na sledeći način:

```
SELECT E.dno, AVG (E.sal)
FROM   Employees E, Departments D
WHERE  E.dno=D.dno AND D.mgrname='Robinson'
GROUP BY E.dno
```

U poslednjem primeru se može koristiti index-only plan, tako da se i ne traže torke iz relacije, što nije slučaj u prethodnom primeru.