Evaluacija relacijskih operacij

Iztok Savnik, FAMNIT

Prosojnice & učbenik

- Učbenik:
  - Raghu Ramakrishnan, Johannes Gehrke, *Database Management Systems, McGraw-Hill, 3rd ed., 2007.*
- *Prosojnice:*
  - *From „Cow Book“: R.Ramakrishnan, http://pages.cs.wisc.edu/~dbbook/*

Pregled evaluacije vprašanj

- *Plan*: Drevo *R.A. op., z izborom alg. za impl. operacij*
  - Vsak operator je tipično implementiran s `pull' vmesnikom: iz operatorja 'povlečemo' naslednji zapis, le-ta povleče zapise iz vhodov (parametrov).
- Dve pomembni temi optimizacije vprašanj:
  - Katere plane dane poizvedbe pregledamo?
    - Algoritem za preiskovanje iskalnega prostora za iskanje najcenejšega (ocena) plana.
  - Kako ocenimo plan pozvedbe?
- Idealno: Želimo najti najboljši plan. Praktično: Izogibanje najslabšim planom!
- Študirali bomo pristop System R.

Pogosto uporabljene tehnike

- Algoritmi za evaluacijo rel. operacij pogosto uporabljajo nekaj enostavnih idej:
  - Indeksiranje: Uporaba pogojev iz stavka WHERE za izbiro majhnega št. n-teric pri selekciji in stikih.
  - Iteracija: Včasih hitreje pregledamo vse n-terice čeprav je na razpolago indeks.
    - Včasih je koristno izvesti iteracijo po podatkovnih vpisih indeksa namesto na sami tabeli.
  - Particije: Velikokrat koristi razdeliti problem na več enakih delov – s tem zamenjamo izvajanje časovno potratnih operacij s podobnimi operacijami nad manjšim številom n-teric.
  - Sortiranje: Velikokrat je koristno podatke najprej sortirati. Sortirane podatke se uporabi pri implementaciji več relacijskih operacij.

** Bodimo pozorni na te tehnike pri opisu izvajanja operacij!*

Statistika in katalogi

- Potrebujemo podatke o relacijah in indeksih. *Katalogi* tipično vsebujejo vsaj:
  - # zapisov (NTuples) in # strani (NPages) za vsako relacijo.
  - # različnih vrednosti ključa (NKeys) in št. strani (NPages) za vsak indeks.
  - Višina indeksa, najmanjši/največji ključ (Low/High) za vsak drevesni indeks.
- Katalogi se periodično popravljajo.
  - Popravljanje ob vsaki spremembi je predrago; uporablja se veliko približnih ocen, manjša nekonsistenca je ok.
- Več podrobnosti (npr. histogrami vrednosti atributov) je včasih shranjeno.

Dostopna pot

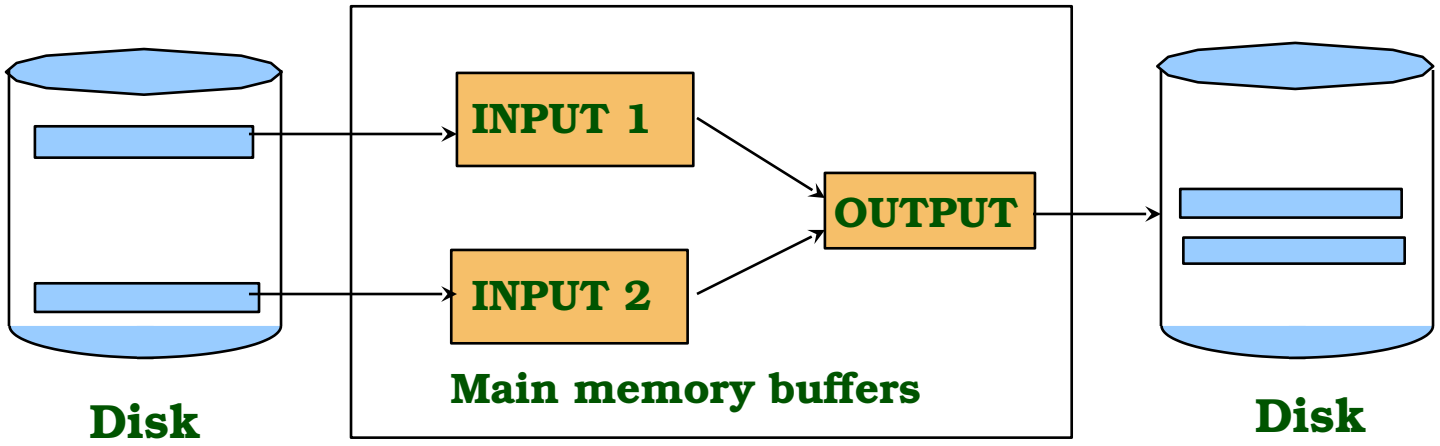
- Dostopna pot je metoda za branje zapisov:
  - Preged datoteke, ali indeksa, ki se ujema s selekcijo (v poizvedbi)
- Drevesni indeks *se ujema* (s konjunkcijo pogojev) z izrazi, ki vsebujejo atr. iz prefiksa isk. ključa.
  - Npr., drevesni indeks na *<a, b, c>* se ujema ts selekcijo *a=5 AND b=3*, in *a=5 AND b>6*, ne pa z *b=3*.
- Razpršilni indeks *se ujema* (s konjunkcijo pogojev) z izrazi, ki vsebujejo pogoj *attribute=value* za vsak atribut iskalnega ključa indeksa.
  - Npr., razpršilni indeks na *<a, b, c>* se ujema z *a=5 AND b=3 AND c=5*; na ujema pa se z *b=3, ali z a=5 AND b=3, ali z a>5 AND b=3 AND c=5*.

Zakaj sortiranje?

- Klasični problem v računalništvu.
- Podatke zahtevamo v urejenem vrstnem redu.
  - Npr., poišči študente po narašč. vrstnem redu *povprecna_ocena.*
- Sortiranje je prvi korak masovnega nalaganja B+ drevesa.
- Sortiranje se uporablja pri izločanju duplikatov. (Zakaj?)
- *Stik uredi-zlij* uporablja sortiranje.
- Problem: sortiraj 1Gb podatkov z 1Mb RAM.
  - Zakaj ne virtualni spomin?

Dvosmerno sortiranje: 3 strani

- Prehod 1: Preberi stran, jo sortiraj, in jo napiši.
  - Uporabimo samo eno stran vmesnega pomnilnika.
- Prehodi 2, 3, …, itd.:
  - Uporabi tri strani vmesnega pomnilnika..



Dvosmerno zunanje sortiranje z zlivanjem

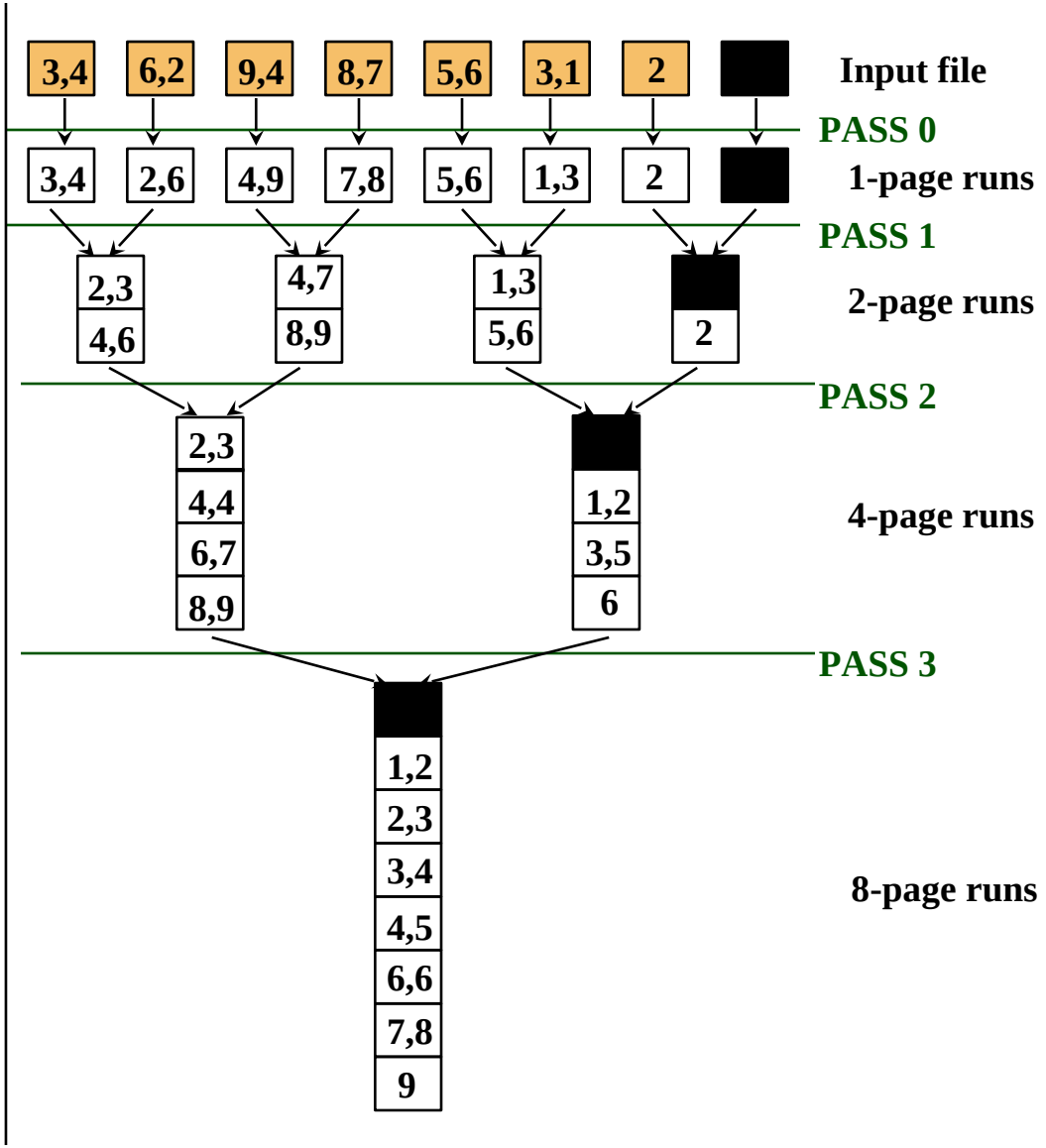
- V vsakem prehodu preberemo + zapišemo vsako stran datoteke
- N strani datoteke => število prehodov

$$= \lceil \log_2 N \rceil + 1$$

- Celotna cena:

$$2N\left(\lceil \log_2 N \rceil + 1\right)$$

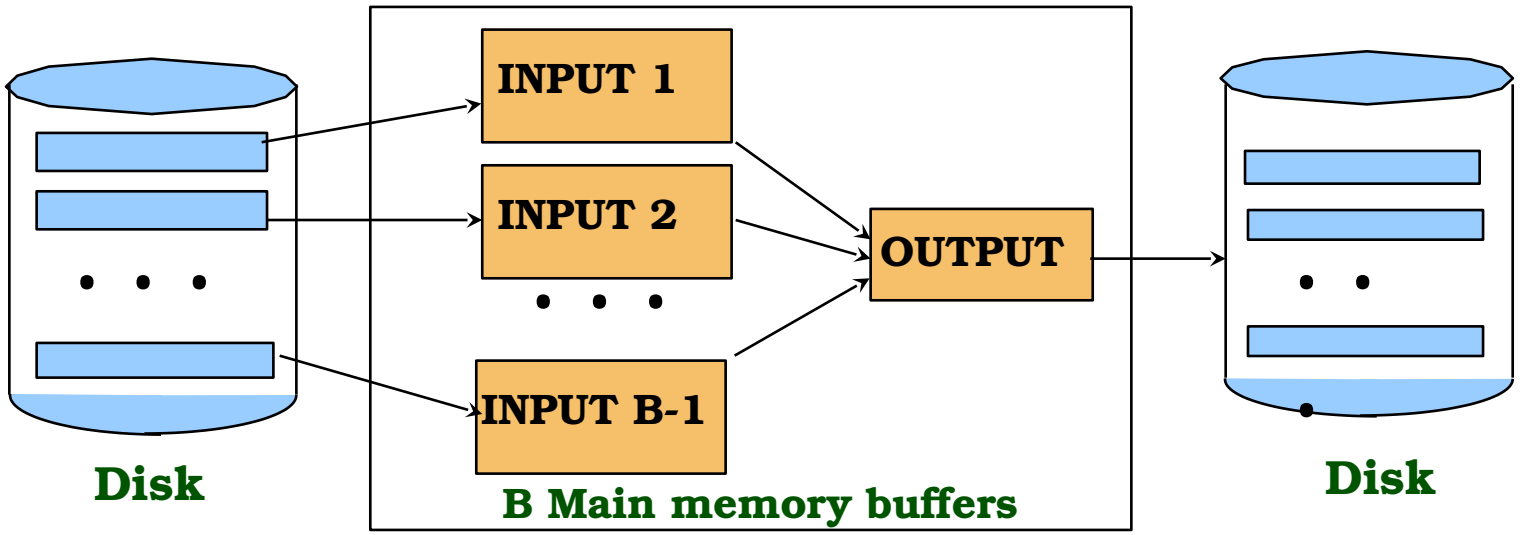
- *Ideja:*
  - *Deli in vladaj:*
  - sortiraj pod-datoteke in zlij skupaj rezultate.



Splošno zunanje sortiranje z zlivanjem

- *Imamo več kot 3 strani izravnalnika.*
- *Kako jih lahko uporabimo?*

- Sortiranje datoteke z N stranmi in uporaba *B* strani izravnalnika:
  - Prehod 0: uporabi *B* strani izravnalnika. Naredi $\lceil N/B \rceil$ urejenih vrst dolžine *B* strani.
  - Prehod 1, …, itd.: zlij *B-1* vrst.



Cena zunanjega sortiranja z zlivanjem

- Število prehodov: $1 + \lceil \log_{B-1} \lceil N / B \rceil \rceil$
- Cena = 2N * (# prehodov)
- Npr., 5 strani izravnalnika, sortiramo dat. velikosti 108 strani:
  - Prehod 0: $\lceil 108 / 5 \rceil$ = 22 sortiranih vrst po 5 strani (zadnja vrsta ima samo 3 strani)
  - Prehod 1: $\lceil 22/4 \rceil$ = 6 sortiranih vrst po 20 strani (zadnja vrsta ima 8 strani)
  - Prehod 2: 2 sortirani vrsti, 80 strani in 28 strani
  - Prehod 3: Sortirana datoteka velikosti 108 strani

Število prehodov zunanjega sortiranja

N	B=3	B=5	B=9	B=17	B=129	B=257
100	7	4	3	2	1	1
1,000	10	5	4	3	2	2
10,000	13	7	5	4	2	2
100,000	17	9	6	5	3	3
1,000,000	20	10	7	5	3	3
10,000,000	23	12	8	6	4	3
100,000,000	26	14	9	7	4	4
1,000,000,000	30	15	10	8	5	4

Relacijske operacije

- Ogledali si bomo implementacije
  - Selekcija (σ)
  - Projekcija (π)
  - Stik (⋈)
  - Razlika (−)
  - Unija (∪)
  - Agregacijske op. (SUM, MIN , itd.) in GROUP-BY
- RA je abstrakten funkcijski jezik
  - Operacije vračajo relacije ⇒ lahko jih sestavljamo!
  - Funkcije višjega reda, kompozicija operacij
- Ko obdelamo operacije bo predstavljena optimizacija
  - Kako optimalno sestavimo skupaj operacije vprašanja?

Shema za primere

Mornarji (*mid*: integer, *mime*: string, *ocena*: integer, *star*: real)
Rezervacije (*mid*: integer, *lid*: integer, *dan*: date, *rime*: string)

- Podobno kot stara shema; dodan *rime*
- Rezervacije:
  - Vsaka n-terica ima 40 zlogov, 100 n-teric na stran, 1000 strani.
- Mornarji:
  - Vsaka n-terica ima 50 zlogov, 80 n-teric na stran, 500 strani.

Komentar o kompleksni selekciji

(dan<8/9/94 AND mime='Janez') OR lid=5 OR mid=3

- Izbirni pogoj je najprej preveden v *konjuktivno normalno obliko* (KNO):

  (dan<8/9/94 OR lid=5 OR mid=3) AND (mime='Janez' OR lid=5 OR mid=3)
- Obravnavamo samo primere brez OR;

  knjiga predstavlja tudi bolj splošne primere.

Uporaba indeksov za selekcijo

```
SELECT  *
FROM    Rezerv R
WHERE   R.rime < 'C%'
```

- Cena je odvisna od # izbranih zapisov in od povezanosti indksa:
  - Cena iskanja izbranih podatkovnih vpisov (tipično majhna). Cena branja zapisov je lahko velika brez povezanosti indeksa.
  - Primer: *rime* < *"C%"* (Rezervacije). Če predpostavimo enakomerno porazdelitev potem 10% zapisov ustreza pogoju (100 strani, 10000 zapisov). S povezanim indeksom je cena nekaj več kot 100 V/I; če pa ni povezan pa do 10000 V/I!
- *Pomembna izboljšava za nepovezane indekse*:
  1. Poišči izbrane podatkovne vpise
  2. Sortiraj rid-je podatkovnih zapisov, ki se bodo brali.
  3. Preberi rid-je po vrsti. To zagotavlja, da je vsaka podatkovna stran prebrana samo enkrat.
  - # takšnih strani je verjetno večje kot pri povezanem indeksu.

Dva pristopa k splošni selekciji

- Prvi pristop: Poišči *najbolj selektivne dostopne poti*, preberi zapise z njim in uporabi preostale iskalne pogoje, ki se ne ujemajo z indeksom:
  - *Najbolj selektivna dostopna pot:* Indeksni dostop ali pregled datoteke za katerega ocenimo, da bo rabil najmanj I/O operacij.
  - Izrazi, ki se ujemajo z indeksom zmanjšajo št. prebranih n-teric; preostali pogoji izločijo nekatere od prebranih n-teric; ne vplivajo pa na št. prebranih strani.
  - Pogoj *dan<8/9/94 AND lid=5 AND mid=3.* Uporabi B+ drevo s ključem *dan.* Pogoja *lid=5* in m*id=3* je potrebno preveriti za vsako prebrano n-terko. Podobno, lahko uporabimo razp. indeks na <l*id, mid*>; *dan<8/9/94* je potrebno še preveriti.

Presek rid-jev

- Drugi pristop: če imamo na razpolago dva ali več indeksov z alternativama (2) ali (3) za pod. vpise:
  - Preberi množice rid-jev podatkovnh zapisov z uporabo primernega indeksa.
  - Potem naredi *presek* množic rid-jev.
  - Preberi zapise in uporabi preostale pogoje izbire.
  - Poglejmo si pogoj *dan<8/9/94 AND lid=5 AND mid=3.*
    - Če imamo B+ drevo na atributu *dan* in razpršilni indeks na *mid*, oba uporabljata alternativo (2).
    - Poiščemo lahko:
      - rid-je zapisov, ki zadoščajo *dan<8/9/94,*
      - *rid-je zapisov, ki zadoščajo mid=3*,
      - naredimo presek in preverimo *lid=5.*

Projekcija

```
SELECT   DISTINCT R.mid, R.lid
FROM     Rezervacije R
```

- Najdražji del je odstranitev duplikatov.
  - SQL sistemi ne odstranijo duplikate, če ne dodaš ključne besede DISTINCT.
- Pristop s sortiranjem: Sortiraj po <mid,lid> in odstrani duplikate. (Lahko optimiziramo tako, da odstranimo nepotrebne zapise (duplikate) med sortiranjem.)
- Pristop z razprševanjem: Kreiraj particije z razpršilno funkcijo na <mid,lid>. Naloži particije v spomin eno po eno, zgradi razpršilo tabelo v spominu ter odstrani duplikate.
- Če obstaja indeks s ključem, ki vsebuje R.mid in R.lid je mogoče ceneje sortirati podatkovne vpise.

Projekcija s sortiranjem

```
SELECT   DISTINCT R.mid, R.lid
FROM     Rezervacije R
```

- Uporaba zunanjega sortiranja:
  - Spremeni Prehod 0 zunanjega sortiranja tako, da se odstrani nepotrebne atribute. Kreirajo se vrste dolžine B (2B z opt.) strani vendar so zapisi manjši kot vhodni zapisi. (Odvisno od velikosti in števila izpuščenih atributov.)
  - Spremeni fazo zlivanja tako, da se izločijo duplikati. # izhodnih n-teric je torej manjše od vhoda. (Razlika je odvisna od # duplikatov.)
  - Cena: Med Prehodom 0, preberi originalno relacijo (velikost M) in izpiši manjše n-terke. V fazi zlivanja se zmanjša število n-teric v vsakem prehodu. V primeru Rezervacij se začetnih 1000 strani reducira na 250 v Prehodu 0, če je razmerje velikosti 0.25.

Projekcija z razprševanjem

- Faza particioniranja: Preberi R z uporabo ene strani vmesnika. Za vsak zapis, odstrani neželjena polja in apliciraj razpršilno funkcijo h1 za izbiro enega izmed B-1 izhodnih strani.
  - Rezultat je B-1 particij. 2 n-terici iz različnih particij sta sigurno različni.
- Faza odstranjevanja duplikatov: Preberi vsako particijo in zgradi razpršilno tabelo v spominu z uporabo razpršilne fun h2 (<>h1) na vseh poljih. Med branjem izločaj duplikate.
  - Če particija ne gre v spomin lahko apliciramo algoritem rekurzivno na tej particiji.
- Cena: Za particioniranje, preberi R, izpiši vsako n-terko z manjšim številom polj. Izhod se prebere v naslednji fazi.

Diskusija o projekciji

- Uporaba sortiranja je standardna rešitev; boljše obravnavanje izkrivljenih (angl. skewed) podatkov; rezultat je urejen.
- Če indeks vsebuje vse potrebne atribute potem lahko uporabljamo samo indeks.
  - Apliciranje projekcije na podatkovnih vpisih.
- Če urejen (npr. drevo) indeks vsebuje vse potrebne atribute že v predponi iskalnega ključa je izvajanje še hitrejše:
  - Preberemo podatkovne vpise (indeksni pregled) po vrsti, spustimo neželjene atribute in primerjamo sosedne vpise za izločitev duplikatov.

Stik po enakosti enega atributa

```
SELECT  *
FROM    Rezervacije R1, Mornarji S1
WHERE  R1.mid=S1.mid
```

- Algebra: R ⋈ S. Optimizirati je potrebno previdno! R x S je velika relacija; strategija z R x S ni učinkovita.
- Predpostavke:
  - M strani R, p_R n-teric na stran
  - N strani v S, p_S n-teric na stran
  - R = Rezervacije, S = Mornarji
- Bolj kompliciranе stike bomo obravnavali kasneje
- *Metrika cene*: # V/I blokov, ostalo ignoriramo

Stik z vgnezdenima zankama

foreach tuple r in R do
    foreach tuple s in S do
        if r_i == s_j then add <r, s> to result

- Za vsako n-terico iz zunanje relacije pregledamo celotno notranjo relacijo S.
  - Cena: M + p_R * M * N = 1000 + 100*1000*500 = 50.001.000 V/I.
- Vgnezdeni zanki po straneh:
  - Za vsako stran R preberi vsako stran S in izpiši spete pare n-teric <r, s>, kjer je r iz R-strani in s iz S-strani.
  - Cena: M + M*N = 1000 + 1000*500 = 501.000 V/I.
  - Če je manjša relacija zunanja, potem je cena = 500 + 500*1000 = 500.500 V/I.

Vgnezdeni zanki z indeksom

foreach tuple r in R do
    foreach tuple s in S where r_i == s_j do
        add <r, s> to result

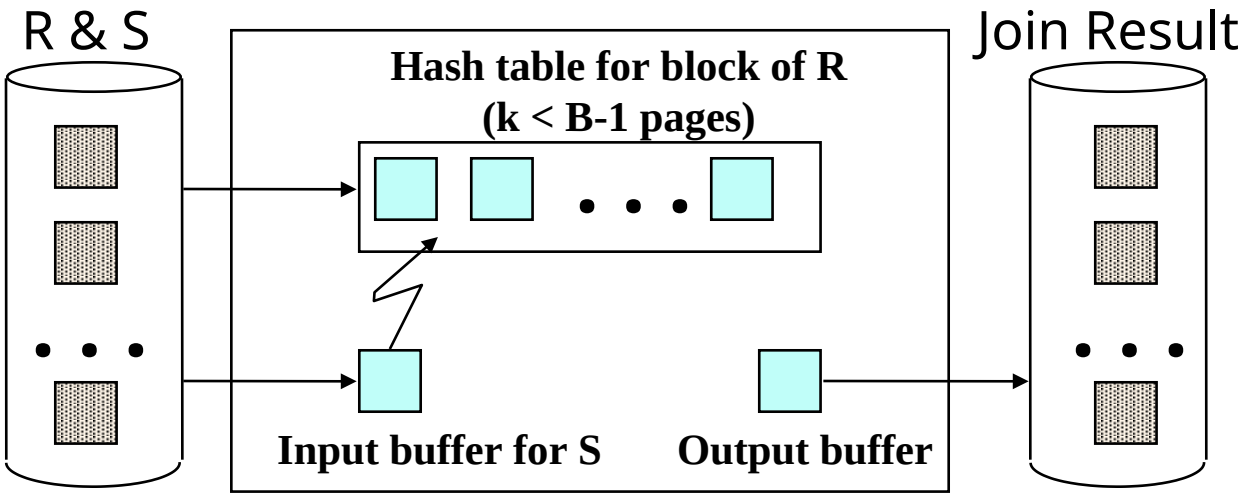
- Če obstaja indeks na eni izmed relacij ga uporabimo za notranjo zanko.
  - Za ∀ n-terico iz R z indeks. poiščemo n-terice, ki se ujemajo v S.
  - Cena: M + ((M*p_R) * cena iskanja n-teric iz S)
- Cena za preverjanje vsake n-terice iz R je branje S:
  - 1.2 strani v primeru razpršilnega indeksa,
  - 2-4 v primeru B+ drevesa.
  - Cena iskanja n-teric v S je v veliki meri odvisna od povezanosti S z relacijo – št. prebranih strani se precej zmanjša.
  - Povezan indeks: 1 V/I (tipično), nepovezan: do 1 V/I za eno S n-terico

Primeri: Vgnezd. zan. z indeksom

- Razpršilni indeks (Alt. 2) na *mid* relacije Mornarji (notranja):
  - Pregled Rezervacije: 1000 strani V/I, 100*1000 n-teric.
  - Za vsako n-terico Rezervacij: 1.2 V/I za branje podatkovnega vpisa + 1 V/I za izbrano n-terico Mornarjev.
  - Cena: 1000 + 1000*100*2.2 = 221,000 V/I.
- Razpršilni indeks (Alt. 2) na *mid* relacije Rezervacije (notranja):
  - Pregled Mornarjev: 500 strani V/I, 80*500 n-teric.
  - Za vsako n-terico Mornarjev: 1.2 V/I za iskanje indeksne strani s podatkovnimi vpisi + cena za branje zapisov Rezervacij .
  - Predpostavljamo enakomerno porazdelitev, 2.5 rezervacij na mornarja (100,000 / 40,000).
  - Cena branja rezervacij je 1 ali 2.5 V/I odvisno od tega ali je indeks povezan.
  - Cena povezan: 500 + 80*500*(1.2+1) = 88.500 V/I.
  - Cena nepovezan: 500 + 80*500*(1.2+2.5) = 148.500 V/I.

Vgnezdeni zanki po blokih

- Uporabi eno stran vmesnika pri pregledu notranje relacije S, eno stran za rezultat in uporabi preostale strani za blok (B-1 str.) iz zunanje relacije R.
  - Z vsako n-terico r iz bloka R, ki se ujema z n-terico s iz S, dodaj <r,s> rezultatu.
  - Potem preberi naslednji blok R, preglej S, itd.



Primeri stika Vgn. zanki po blokih

- Cena: pregled zun. rel. + # zun. blokov * pregled notranje rel.
  - # zunanjih blokov = # strani rel. / velikost zunanjega bloka
- Če vzamemo Rezervacije (R) za zunanjo relacijo; SUPB ima 100 strani izravnalnika:
  - Cena pregleda R je 1000 V/I; vsega skupaj 10 zun.vrst
  - Za vsako zun.vrsto R pregledamo Mornarje (S); 10*500 V/I.
  - Cena = 1000 + 10*500=6000 V/I
  - Če imamo prostora za samo 90 strani R, moramo pregledati S 12x.
- Če vzamemo Mornarje kot zunanjo relacijo (100 strani izrav.):
  - Cena pregleda S je 500 V/I; skupaj 5 zun.vrst.
  - Za vsako zun.vrsto S, pregledamo Rezervacije;   5*1000 V/I.
  - Cena = 500 + 5*1000 = 5500 V/I

Stik uredi-zlivaj

- Uredi R in S glede na atribut stika. Med pregledom tabel izvedi ``združevanje'' glede na atribut stika in izpiši n-terke.
  - Pregleduj R dokler ni trenutni R-zapis >= trenutnem S zapisu, potem pregleduj S dokler ni tr. S zapis >= tr. R zapisu; delaj to dokler ni tren. R zap. = tren. S zap.
  - V tej točki se ujemajo skupina R zapisov in skupina S zapisov z isto vrednostjo atributa stika. Izpiši vse pare <r,s>, ki se ujemajo v atributu stika.
  - Ponovno začni s pregledom R in S.
- R se pregleda enkrat; vsaka skupina v S se pregleda enkrat za vsak R zapis, ki se ujema po atributu stika. (Zelo verjet. se S skupina shrani v vmesni pomnilnik.)

Primer stika uredi-zlivaj

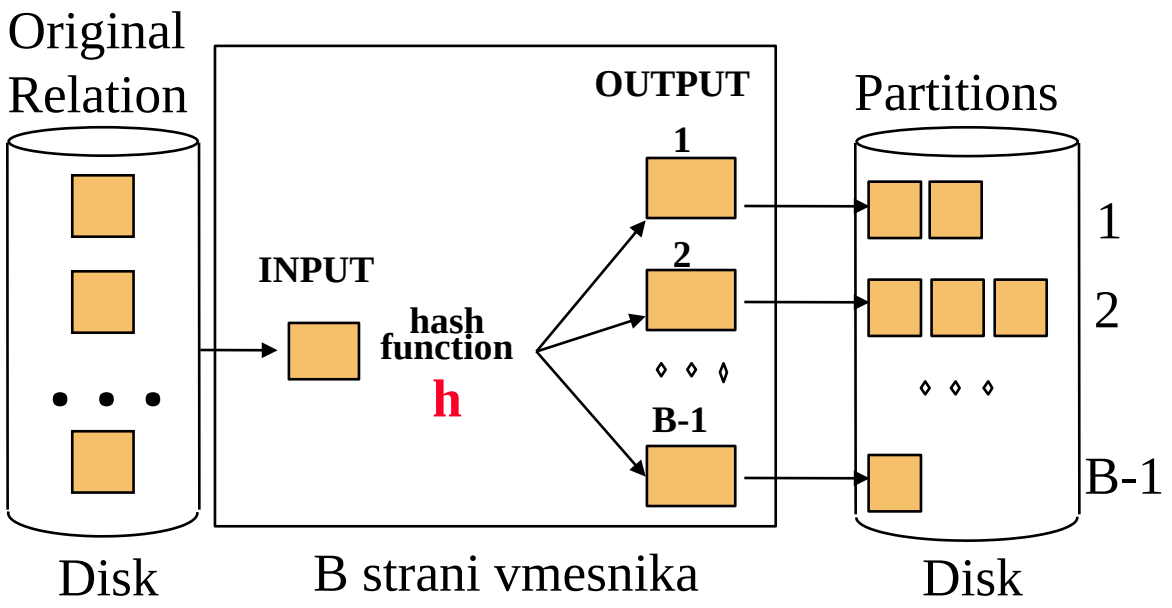
- Cena: M log M + N log N + (M+N)
  - Ocena zlivanja: M + N, lahko tudi M*N (ni zelo verjetno!)
  - Podrobno: $2*M*(1+\log_{B-1}M/B) + 2*N*(1+\log_{B-1}N/B) + M+N$
- Z uporabo 35, 100 ali 300 strani vmesnika lahko tabele Rezervacije in Mornarji sortiramo z 2 prehodi;
- Celotna cena stika: 4*1000+4*500+1000+500=7500 V/I.

Izboljšava stika uredi-zlivaj

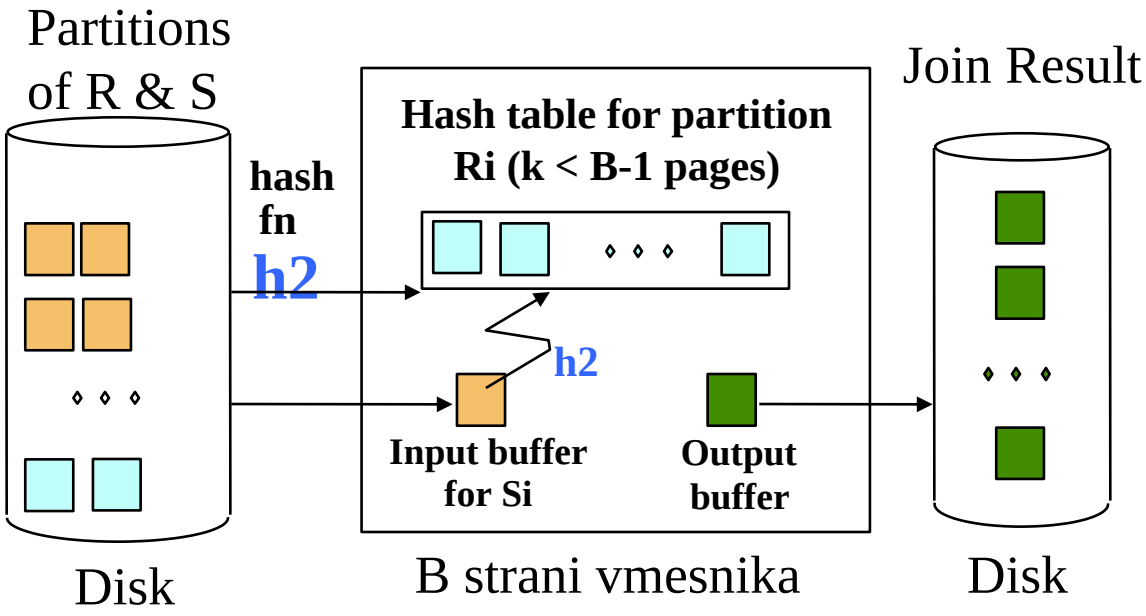
- Kombiniramo lahko faze zlivanja za sortiranje relacij R in S, s fazami zlivanja za implementacijo stika.
  - Če B > √ L, kjer je *L* velikost večje relacije (vrste dolžine B v Prehodu 0), dobimo # vrst vsake relacije < B.
  - Če imamo B > 2*√ L, lahko zlijemo vse vrste obeh relacij z enim branjem relacij. Zasežemo 1 stran za eno vrsto relacije in zlivamo n-terke R in S med preverjanjem pogoja stika.
  - *Izboljšava*: V Prehodu 0 lahko z izboljšavo sort. izdelamo vrste dolžine 2B. Potem je # vrst vsake relacije < B/2 (boljše od < B). Zlivamo lahko 1x več vrst.
  - Cena: branje+pisanje vsake relacije v Prehodu 0 + branje vsake relacije v (edini) fazi zlivanja (pisanje rezultata ne šteje). Skupaj, 3*(M+N)!
  - Naš primer: cena gre dol iz 7500 na 4500 = 3*(1000+500) V/I.
- V praksi je cena stika uredi-zlivaj, podobno kot cena zunanjega sortiranja, linearna.

Razpršilni stik

- Porazdeli obe relaciji z razpršilno funkcijo **h**: n-terice particije i iz R se bodo ujele z n-tericami particije i iz S.



- Preberi particijo R, jo porazdeli z **h2 (<> h!)**. Preglej ustrezno particijo S in poišči n-terice, ki se ujemajo.



Razpršilni stik

- # particij k < B-1 (zakaj?), B-2 > velikosti največje particije, ki bo v spominu. Predpostavljamo enakomerno porazdelitev in maksimiziramo k:
  - k= B-1, in M/(B-1) < B-2, B must be > $\sqrt{M}$
- Zgradimo razpršilno tabelo v spominu za pohitritev ujemanja n-teric (rabimo malce več spomina).
- Če razpršilna funkcija ne porazdeli n-teric enakomerno se lahko zgodi, da nekatere particije ne gredo v spomin.
  - Razprševanje lahko naredimo rekurzivno: R-partiticijo povežemo z ustrezno S-partiticijo.

Cena razpršilnega stika

- Faza izdelave part., branje+pisanje obeh rel., 2(M+N).
  Faza ujemanja, preberi obe rel. = M+N V/I.
- Naš primer, 3 *(1000+500) = 4500 V/I.
- Stik z uredi-zlij vs. Razpršilni stik:
  - Pri minimalni količini spomina imata oba ceno 3(M+N) V/I.
  - Razpršilni stik je superioren, če se velikosti relacij bistveno razlikujejo.
  - Razpršilni stik lahko zelo dobro paraleliziramo.
  - Stik z zlivanjem je manj občutljiv na izkrivljene (angl. skewed) podatke; rezultat je sortiran.

Splošni pogoji stika

- Enakost večih atributov (npr., *R.mid=S.mid* AND *R.rime=S.mime*):
  - Za vgnez.zanko z indeksom kreiraj indeks na <m*id,mime*> (če je S notranja rel.); ali uporabljaj obstoječe indekse na *mid* in *mime*.
  - Za stik sortiranje-zlivanje in razpršilni stik sortiraj/porazdeli s kombinacijo dveh atributov stika.
- Pogoji neenakosti (npr., *R.rime < S.mime*):
  - Za Vgnezdeni zanki z indeksom uporabi (povezano!) B+ drevo.
  - Stik z zlivanjem in razpršilni stik ni uporaben.
  - Stik Vgnezdeni zanki po blokih bo zelo verjetno dal najboljše rezultate.

Operacije nad množicami

- Presek in kartezijski produkt sta posebna primera stika.
- Unija in razlika se implementirata podobno.
- Unija s sortiranjem:
  - Sortiramo obe relaciji (po vseh atributih).
  - Pregled sortiranih relacij in zlivanje
  - *Alternativa*: Zlij vrste iz prehoda 0 iz obeh ralacij.
- Unija z razpršilno funkcijo:
  - Porazdeli R in S z uporabo razpršilne funkcije *h*.
  - Za vsako S-particijo izgradi razpršilno tabelo v spominu (z uporabo h2), preglej ustrezno R-particijo in dodaj n-terice v tabelo medtem, ko se izloča duplIkate.

Agregacijske operacije(AVG,MIN,itd.)

- Brez grupiranja:
  - V splošnem zahteva pregled relacije.
  - Pregled je možen samo z indeksom, če iskalni ključ vsebuje vse atribute v SELECT ali WHERE stavku.
- Z grupiranjem:
  - Sortiraj po group-by atributih, potem preglej relacijo in izračunaj agregacuijske funkcije za vsako skupino.
  - Podoben pristop je osnovan na porazdelitvi (razpršitvi) po group-by atributih.
  - Z drevesnim indeksom, ki vsebuje atribute v SELECT, WHERE in GROUP BY stavke lahko naredimo pregled samo na osnovi indeksa.
    - Če group-by atributi tvorijo predpono iskalnega ključa, potem lahko preberemo vpise/zapise v vrstnem redu določenem z group-by.

Vpliv izravnalnika

- Če se več operacij izvaja vzporedno potem je ocenjevanje št. prostih strani v izravnalniku samo predvidevanje.
- Ponavljajoč vzorec dostopa mora biti usklajen s strategijo zamenjave strani v izravnalniku.
  - Notranja relacija se pregleduje ciklično pri enostavni vgnezdeni zanki.
  - Če imamo zadosti strani, da hranimo notranjo relacijo potem zamenjalna strategija ne igra posebne vloge.
  - Sicer pa je najboljša strategija MRU; LRU je najslabša (sekvenčno prelivanje).
  - Ali je zamenjalna strategija pomembna pri vgnezdeni zanki po blokih?
  - Kaj pa pri vgnezdeni zanki z indeksom? Stik uredi-zlij?

Pregled

- Relacijski SUPB:
  - *Poizvedbe so sestavljene iz nekaj osnovnih operacij.*
  - Implementacija teh operacij je skrbno uglašena.
  - Podrobnosti pri implementaciji so pomembne!
- Obstaja več različnih alternativnih implementacij za vsako operacijo; ne obstaja superiorna tehnika za večino operacij.
- Potrebno je pregledati alternative pri implementaciji vsake operacije v poizvedbi.
  - Izberemo najboljšo strategijo na osnovi statističnih podatkov o tabelah.
  - Izbor ene operacije je del celotnega dela optimizacije celotne poizvedbe.