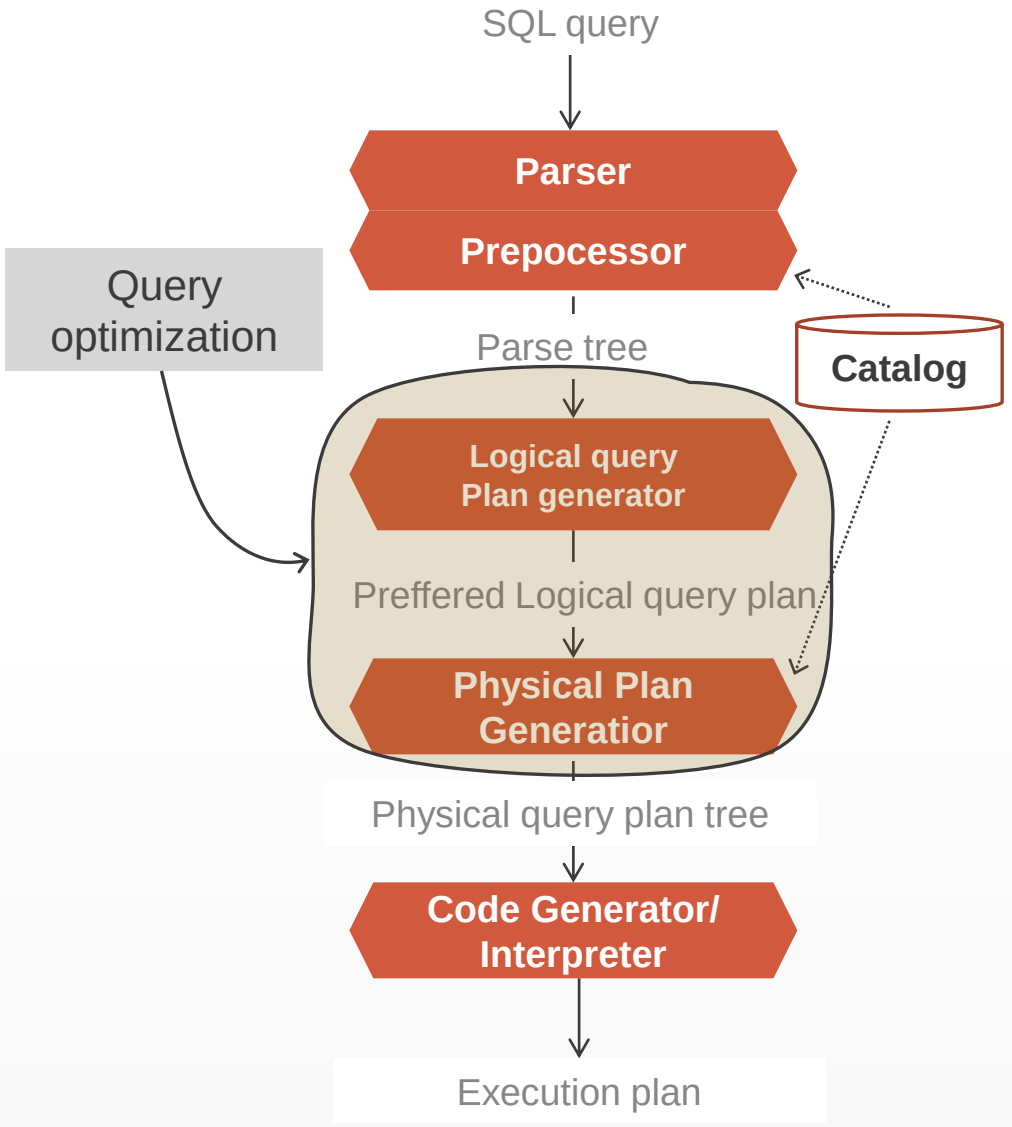
Procesiranje upita

Baze podataka 2

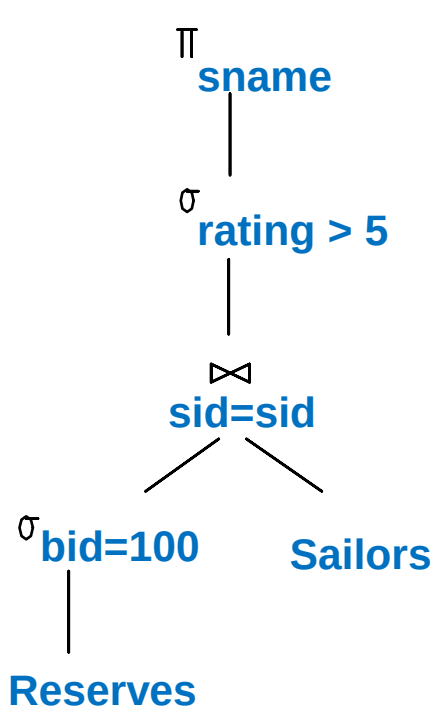
2020/21

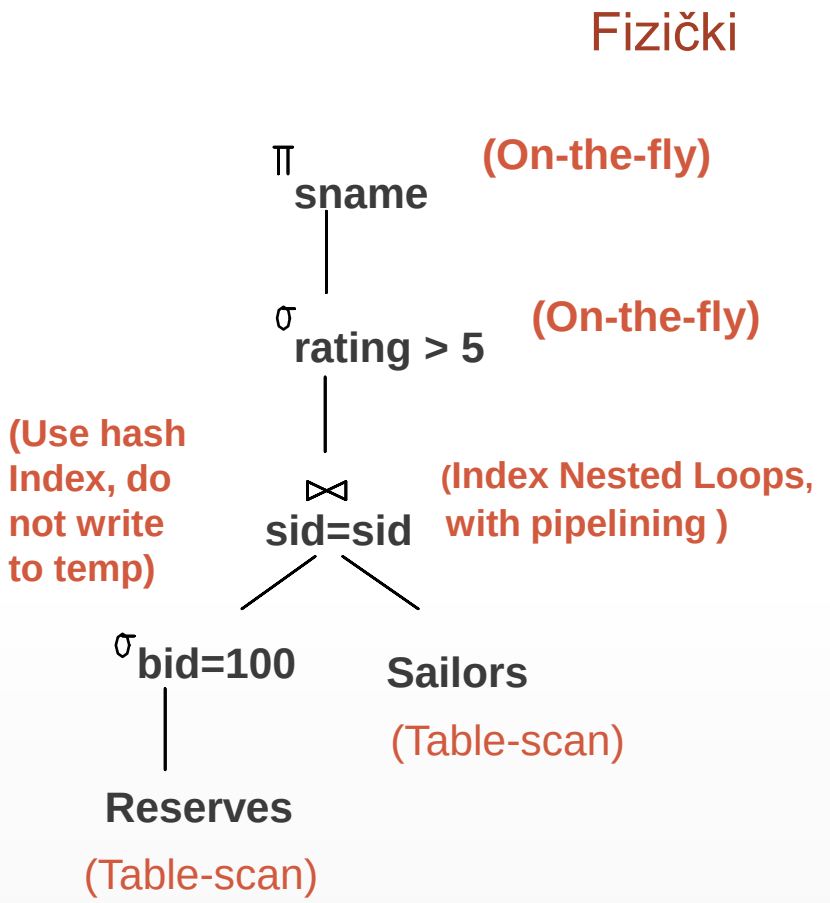
Procesiranje upita

- Obuhvata proces pretveranja upita u niz operacija nad bazom i njihovo izvršavanje.
- Tri osnovna koraka:
  - Parsiranje,
  - Drvo izraza, logički plan upita – *logical query plan*
  - Fizički plan upita - *physical query plan*.



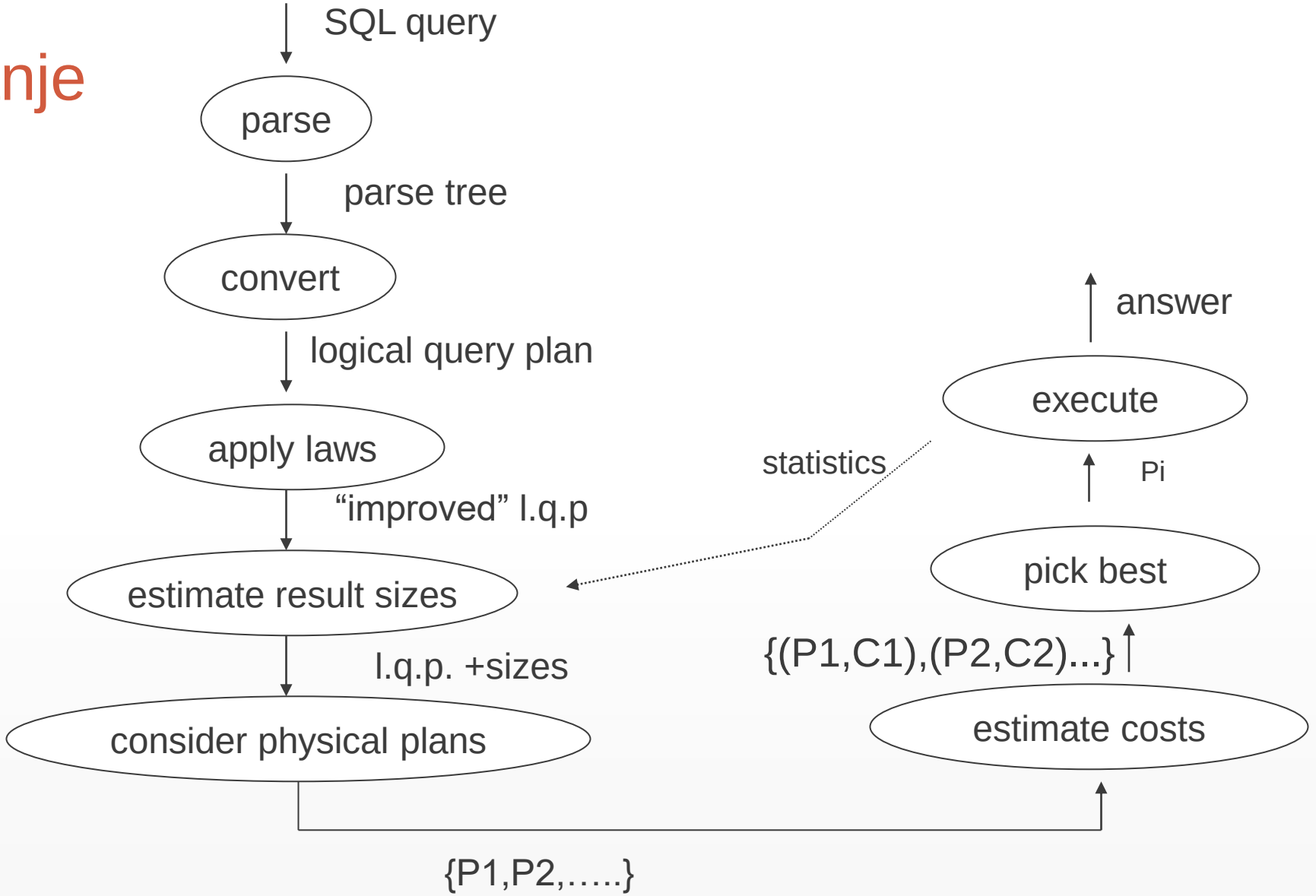
Logički i fizički plan





- Logički plan - stablo proširene relacione algebre
- Fizički plan – logički sa markerima tipa:
  - Način pristupa podacima - *access method*,
  - Algoritam implementacije svake operacije,
  - Sinhronicazija operacija i prosleđivanje međurezultata – *pipeline*, *materialization*.

Procesiranje koraci



Pretraga tabela (*file scan*)

- Tri osnovne tehnike pretrage zajedničke skoro svim operatorima
  - Pretraga putem indeksa (*index scan*)
  - Iterativna/sekvencijalna pretraga fajla (*table scan*)
  - Patricionisanje torki po ključu pretrage.
    Standardne tehnike - sortiranje i heširanje.
  - *Sort-scan* – kada je potrebno dobiti sortirane podatke sa diska; više načina
    - ako postoji indeks po atributu sortitranja
    - ako je tabela dovoljno mala – sortira se po učitavanju memoriju
    - ako je tabela velika - merge-sort algoritam
- Način na koji se dobavljaju torke iz fajla - *access path/method*.
- Access path značajno utiče na cenu izvršavanja svih operatora.

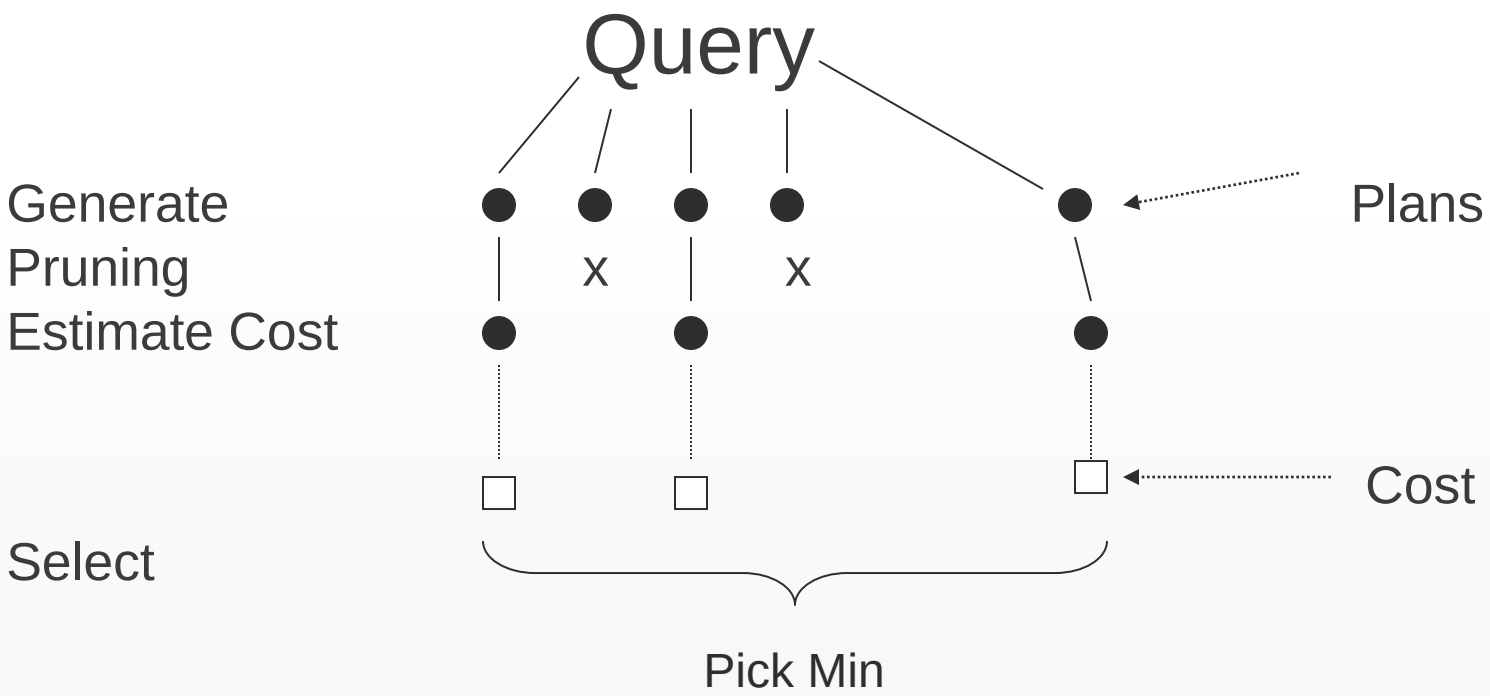
Odluke

- Logički plan
  - Koji se algebarski zakoni primenjuju? - prostor pretrage
  - Koji se logički planovi razmatraju? – pravila optimizacije
  - Kojim redom se radi pretraga prostora? – algoritam optimizacije
- Fizički plan
  - Koje fizičke operatore koristiti?
  - Access paths
  - Pipeline ili materijalizacija

Query optimizers do not “optimize”, but just try to find “reasonably good” evaluation strategies.

Vrste optimizacije

- Heuristički bazirane
- Bazirane na funkcijama troškova



Izvršavanje planova upita

Procesiranje upita

Modeli procesiranja

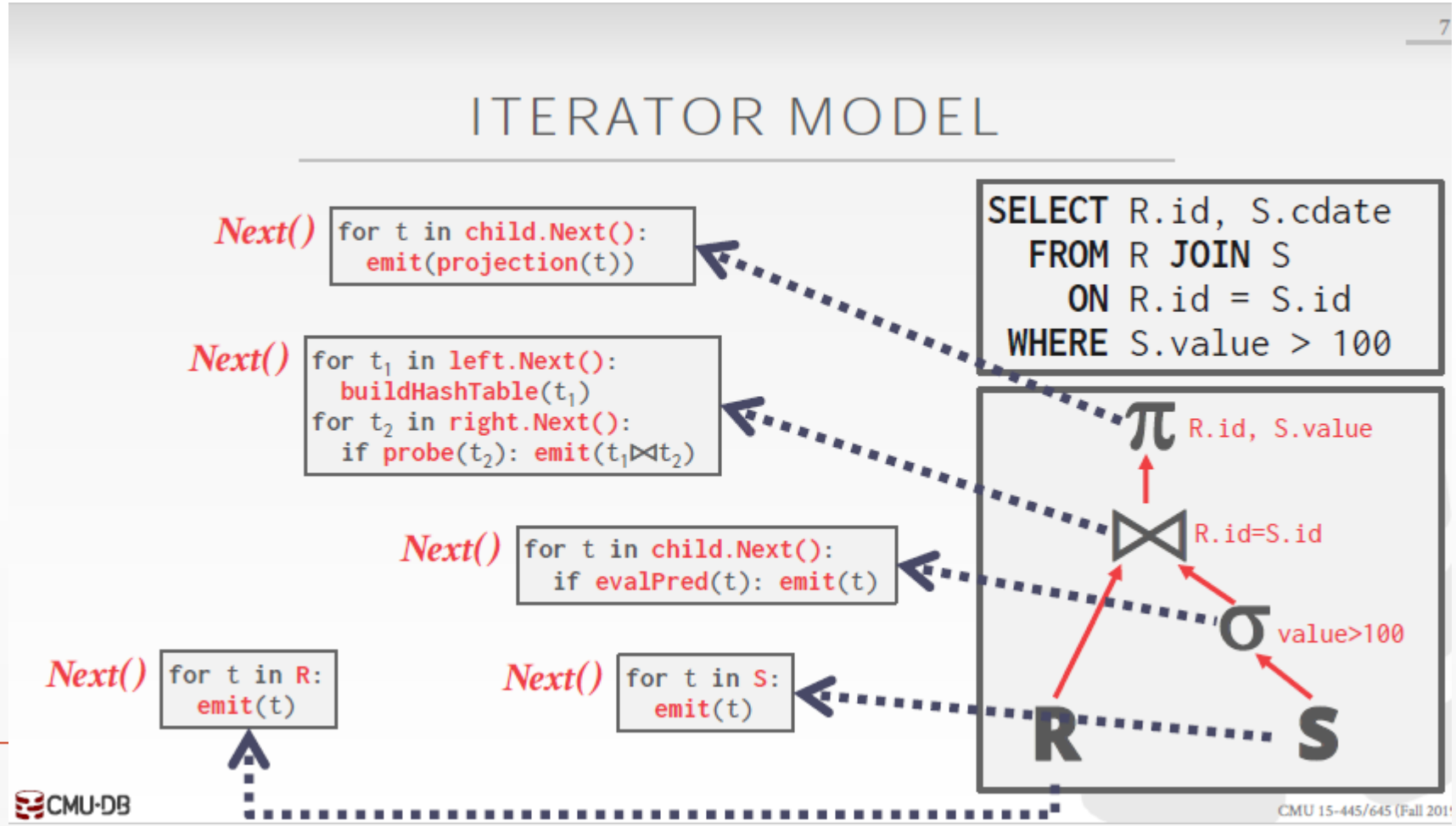
- Kako se izvršava fizički plan upita? Kada se vrši sinhronicazija operacija i prosleđivanje međurezultata?
- Iterisanje
- Materijalizacija
- Batch model

Iteratori za implementaciju fizičkih operatora

- Svaki operator implementira interfejs Iterator.
- Metodi *Iterator*-a:
  - Open()
    - Inicijalizuje stanje iteratora, tj. sve potrebne strukture podataka
    - Setuje parametre, npr. uslov selekcije
  - Get_next()
    - Izvršava se nad ulaznim podacima
    - Izvodi potrebno procesiranja ulaznih podataka i proizvodi izlaz
  - Close()
    - Oslobađa zauzete resurse
- Mogu biti: **Blokirajući** – vraćaju kolekcije torki; **Neblokirajući** – vraćaju jedan po jedan zapis.

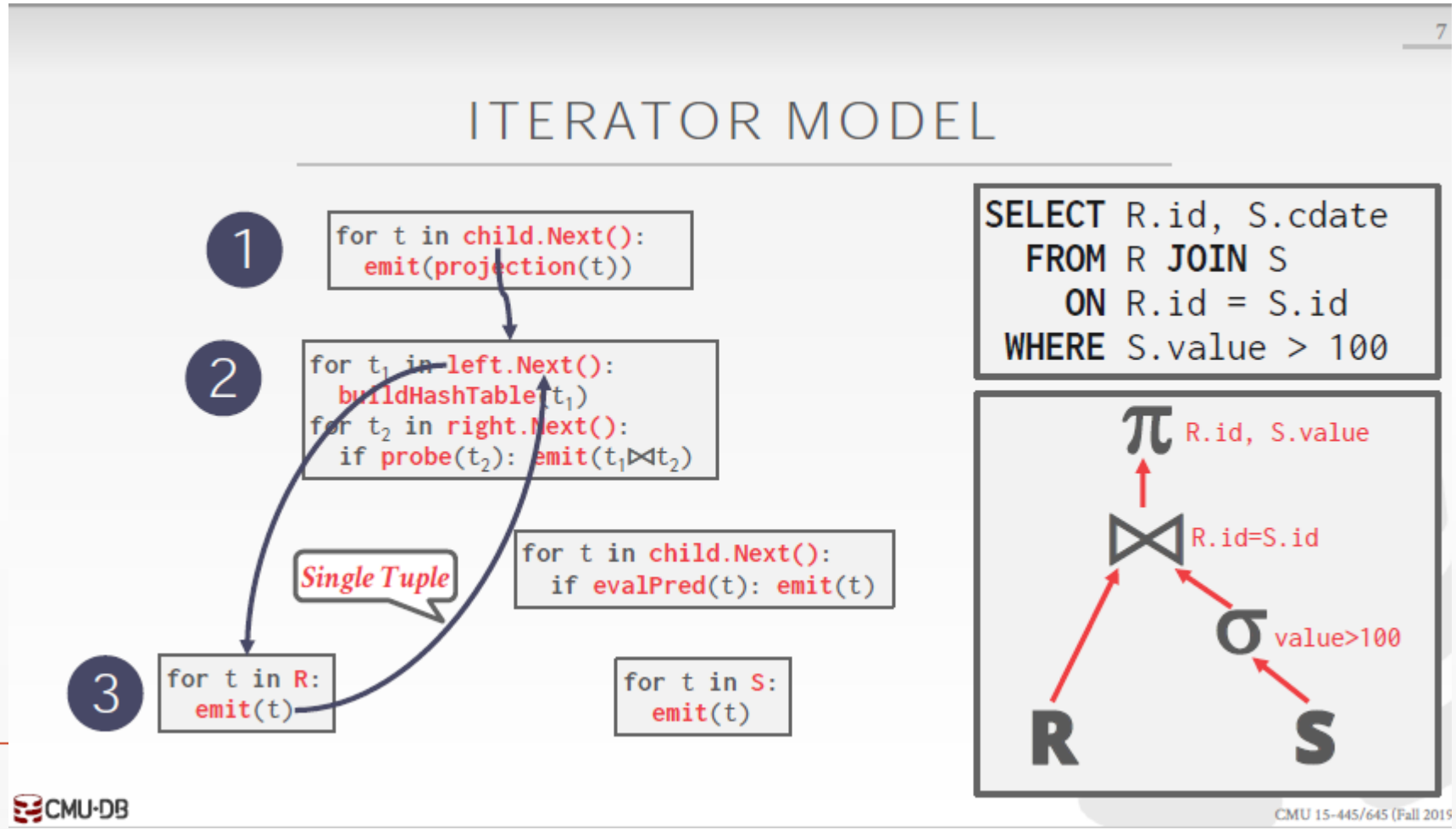
Iteratori (procesiranje iterisanjem)

Volcano ili Pipeline Model
Za neblokirajuće operatore



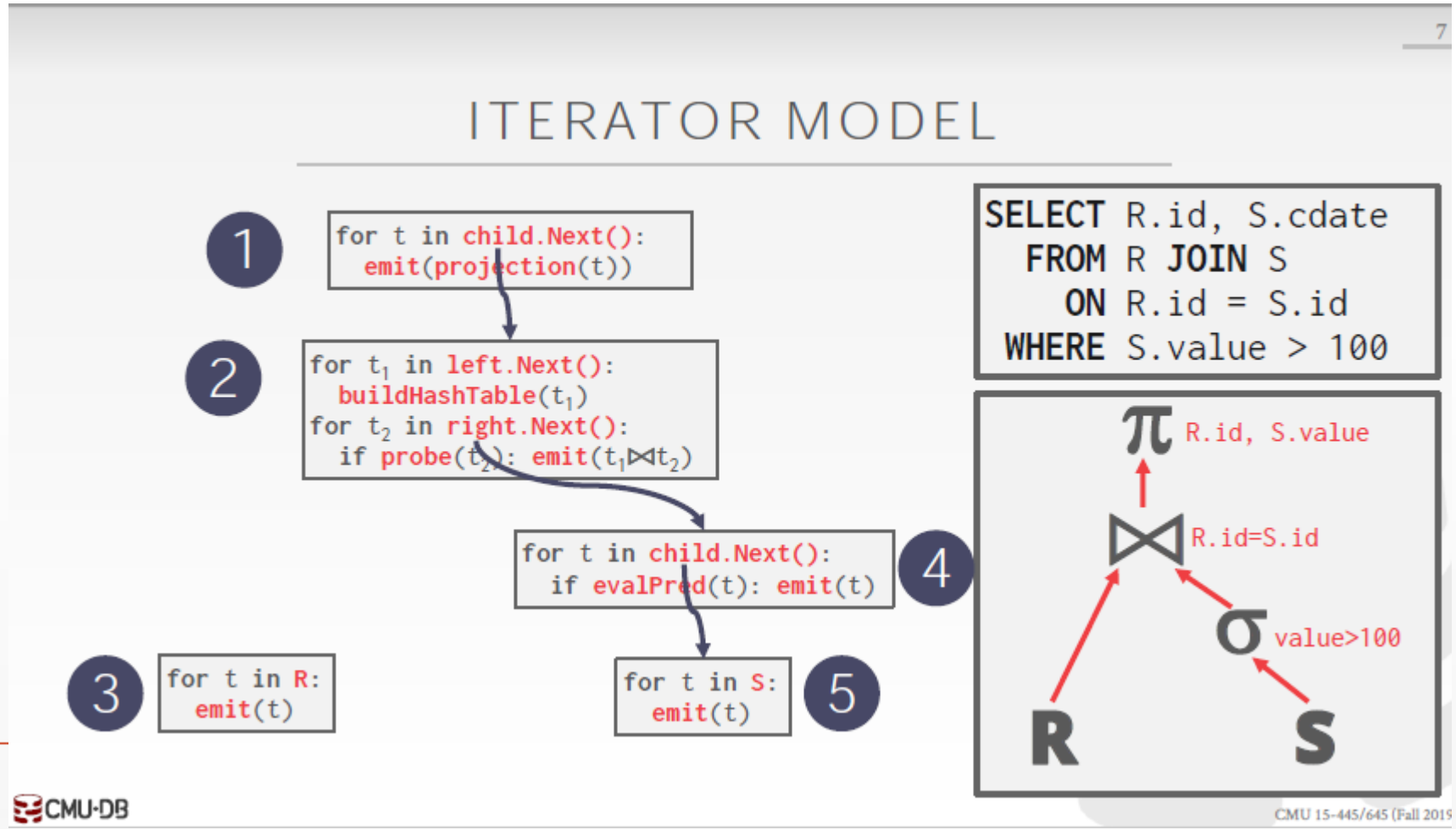
Neblokirajući iteratori (procesiranje iterisanjem)

Volcano ili Pipeline Model



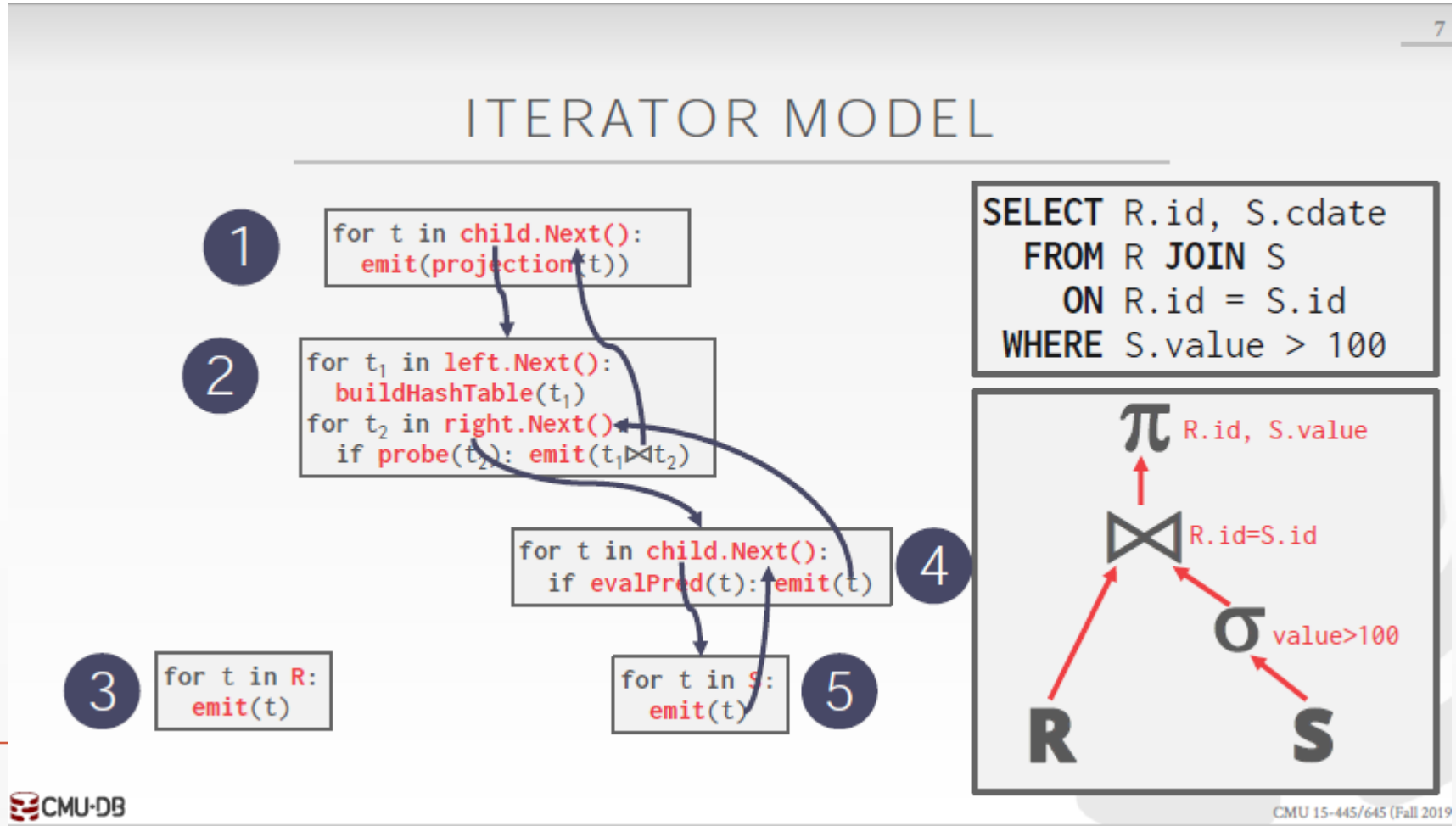
Neblokirajući iteratori (procesiranje iterisanjem)

Volcano ili Pipeline Model



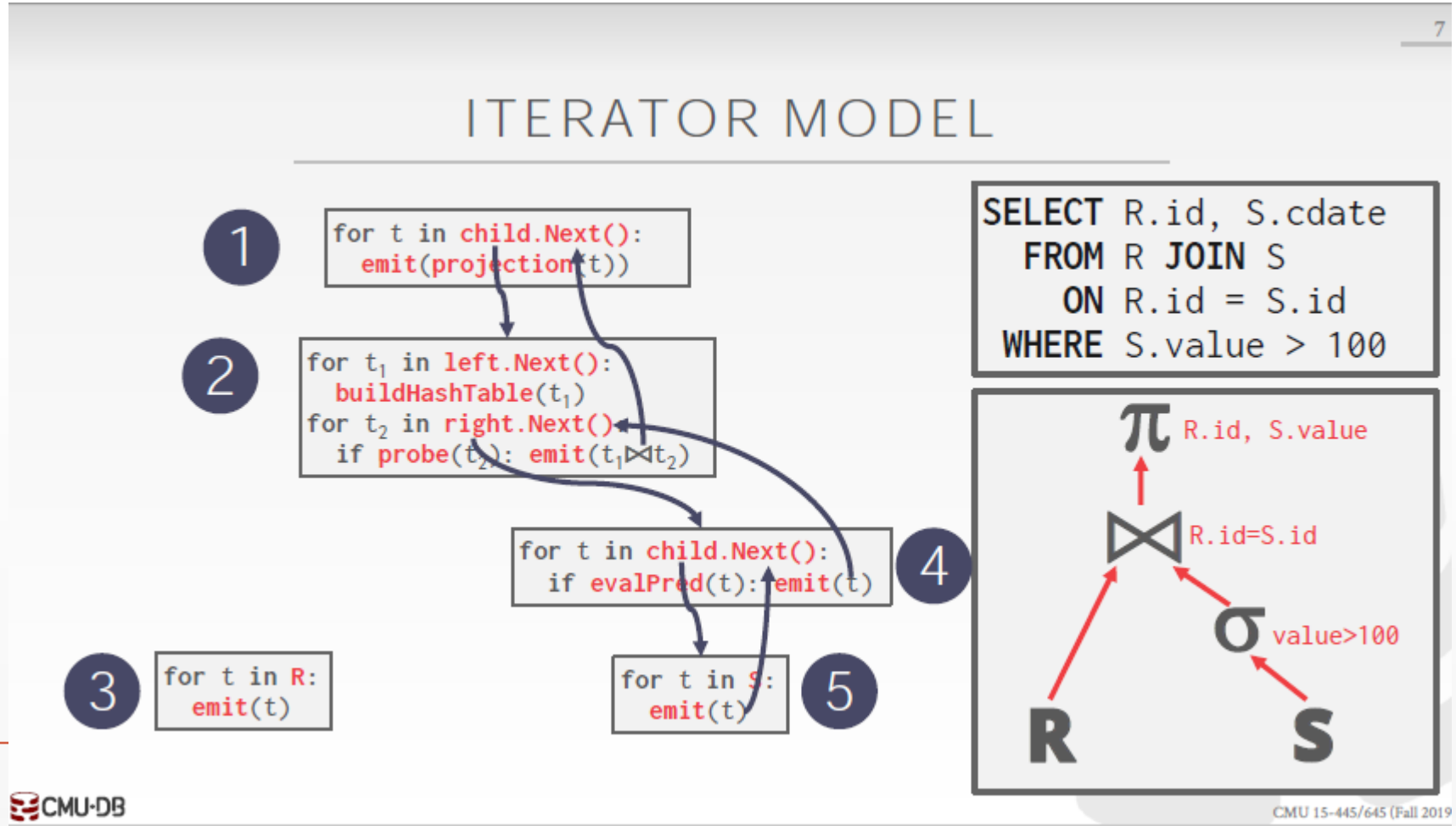
Neblokirajući iteratori (procesiranje iterisanjem)

Volcano ili Pipeline Model



Neblokirajući iteratori (procesiranje iterisanjem)

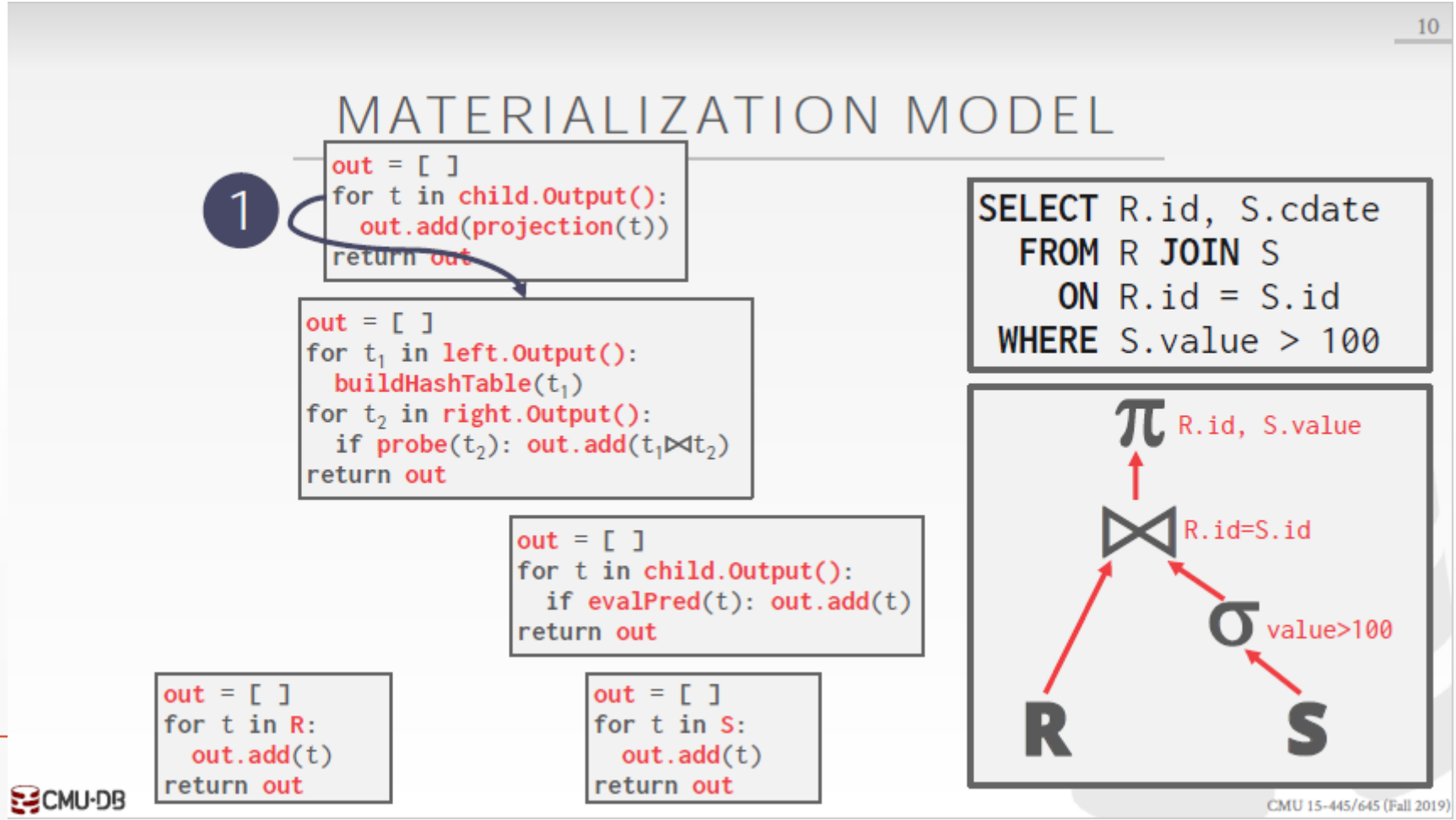
Volcano ili Pipeline Model



Materijalizacija

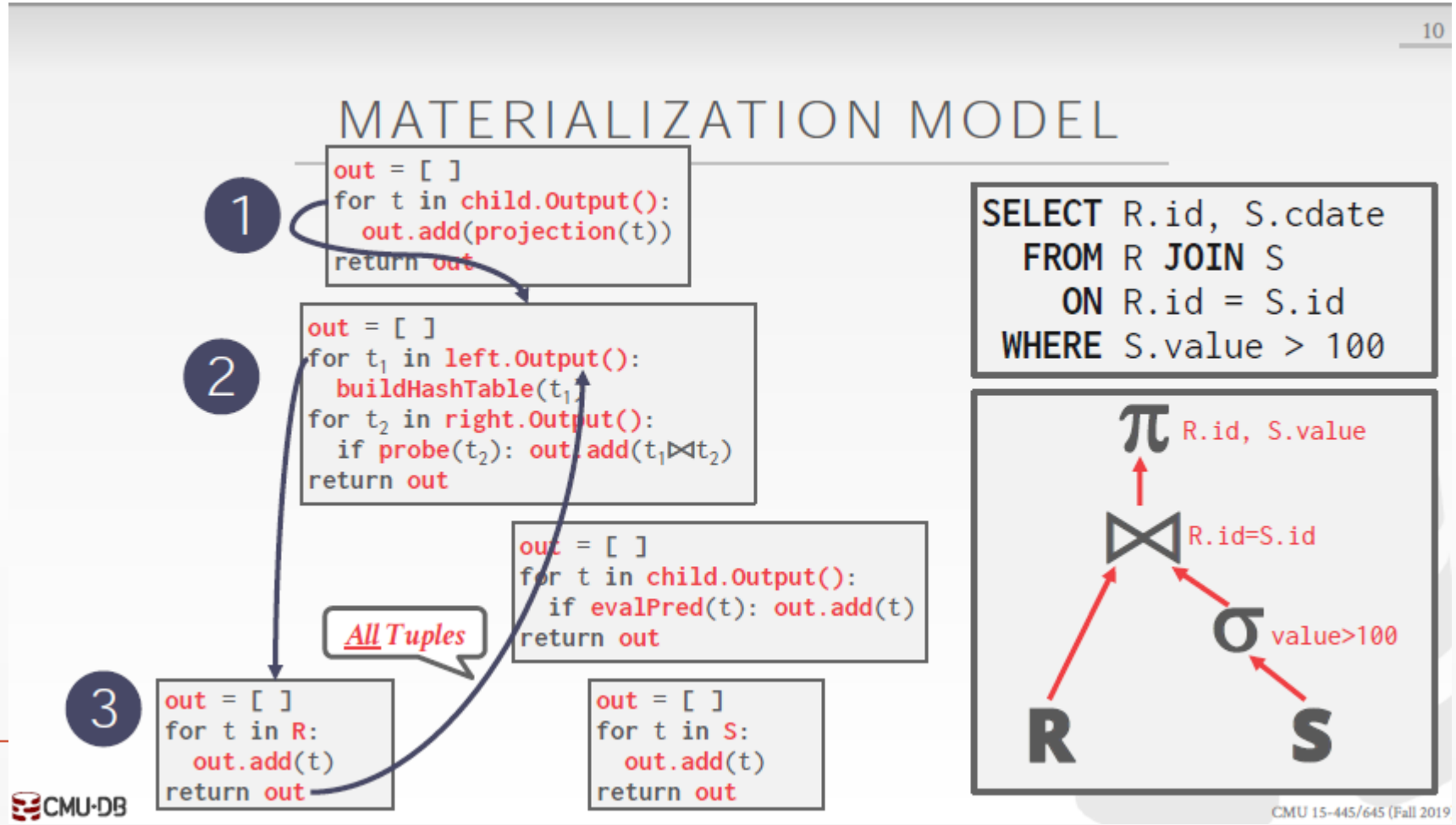
Blokirajući operatori

Ceo rezultat se prosleđuje roditelju odjednom.



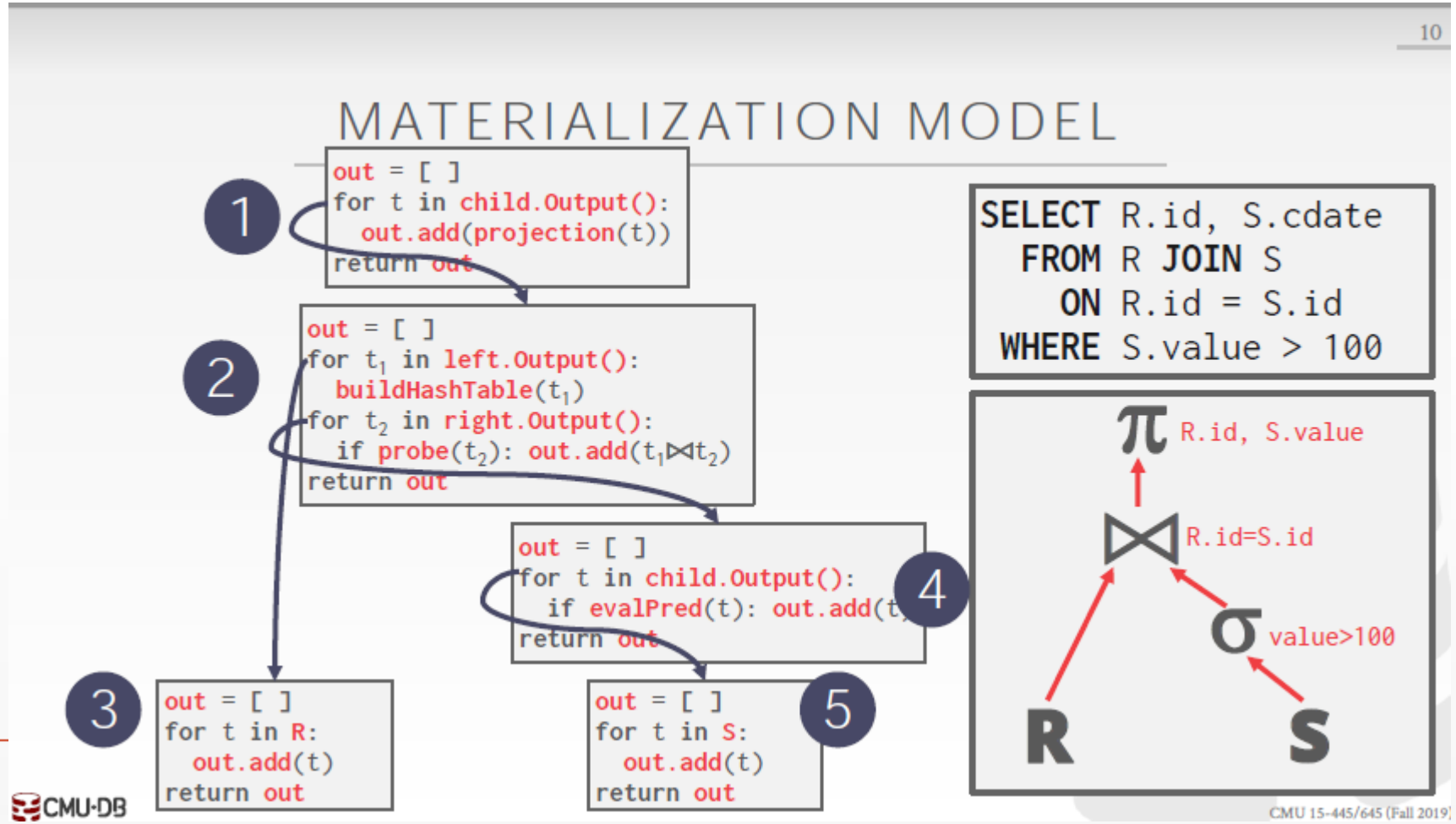
Materijalizacija

Blokirajući operatori



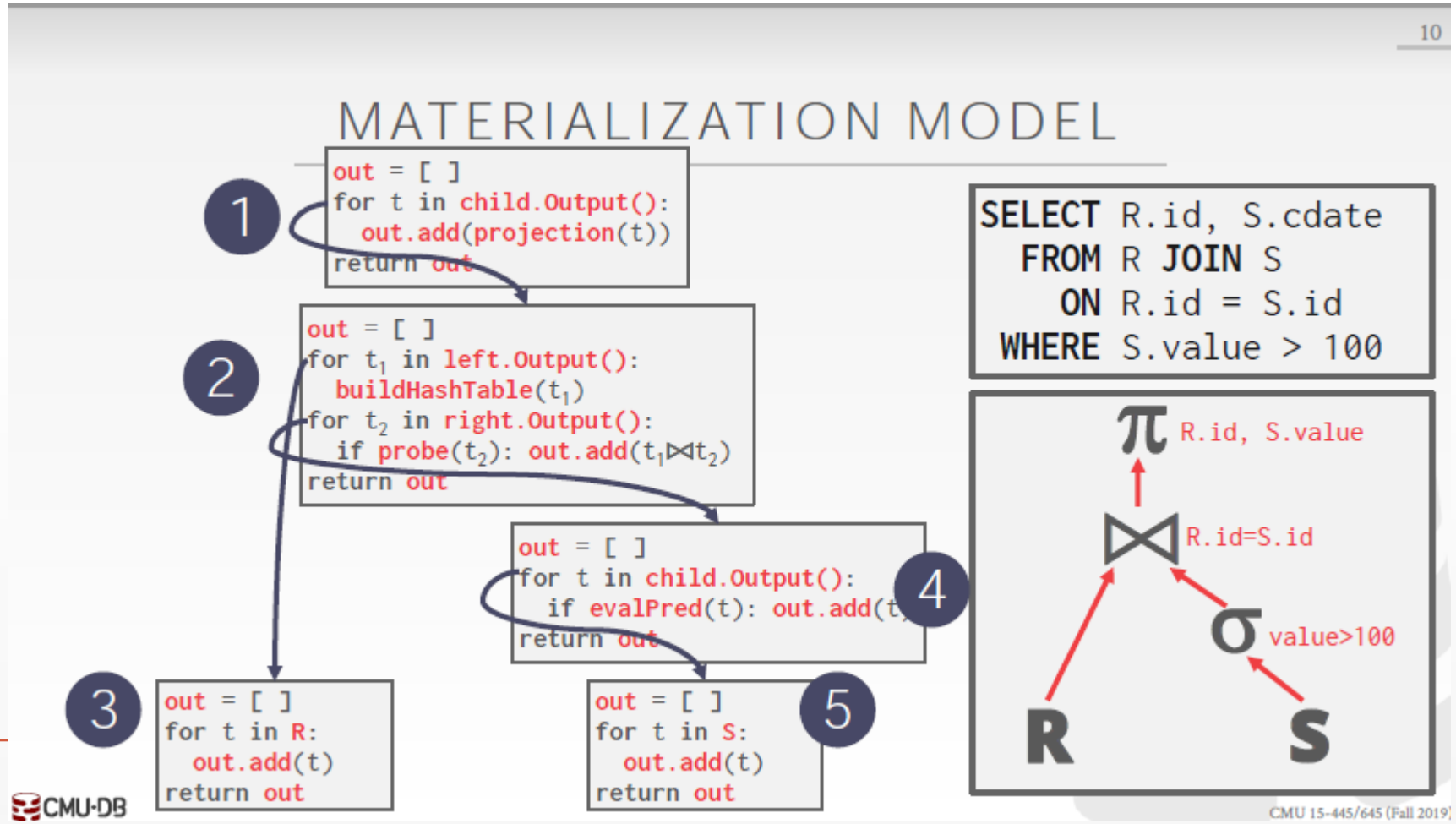
Materijalizacija

Blokirajući operatori



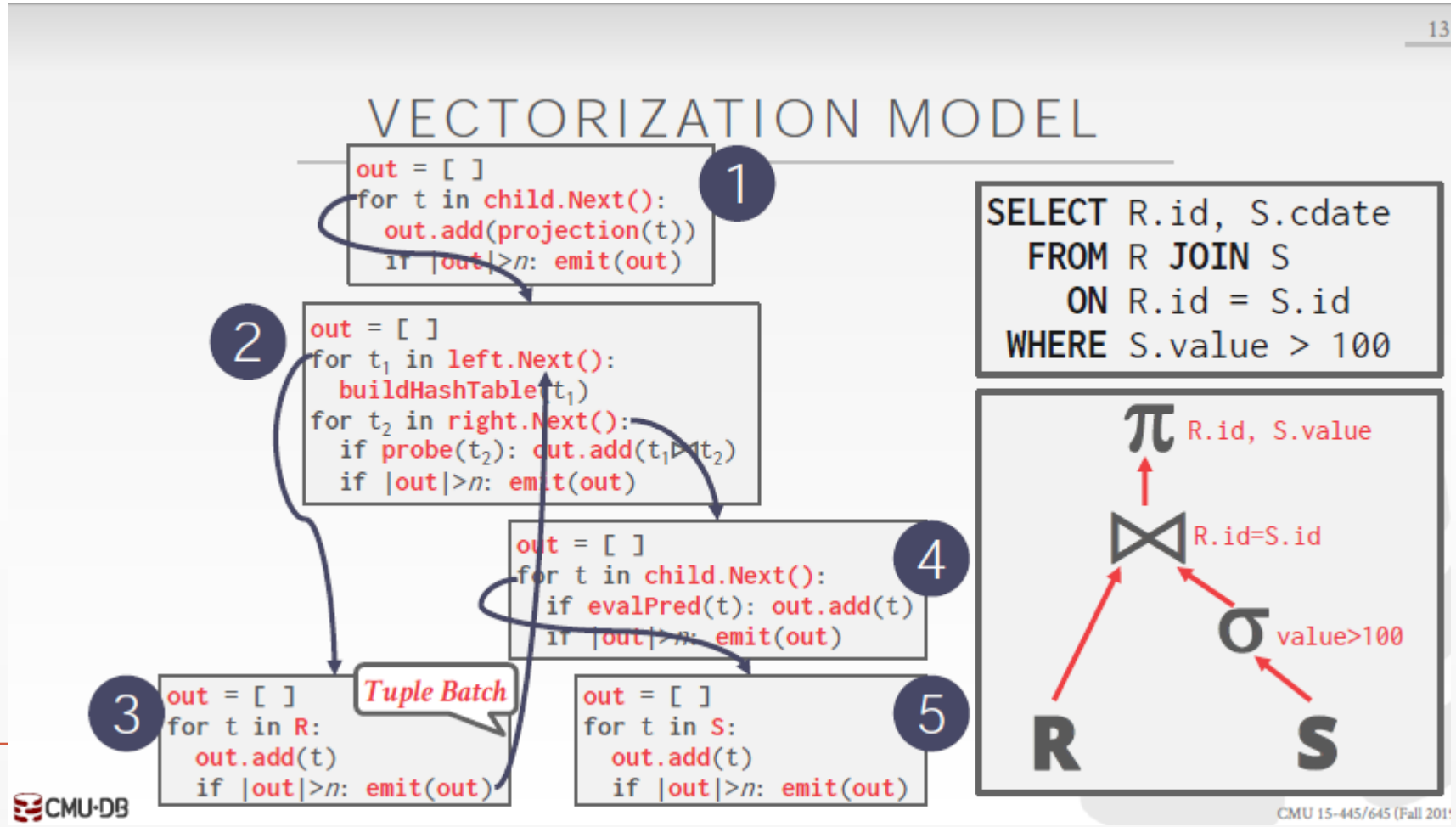
Materijalizacija

Blokirajući operatori



Batch procesiranje

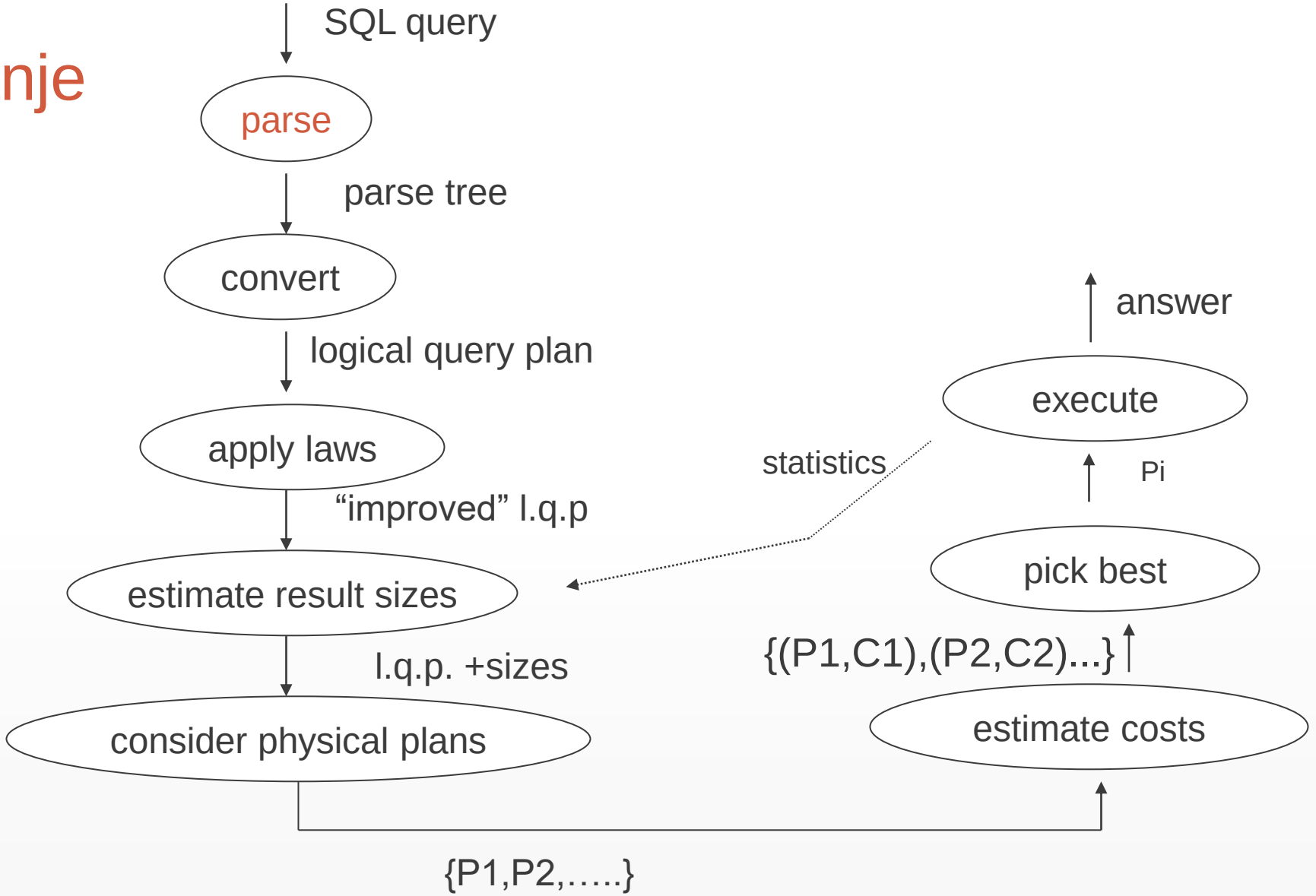
Operator emituje deo po deo rezultata



Koraci – parsiranje i preprocesiranje

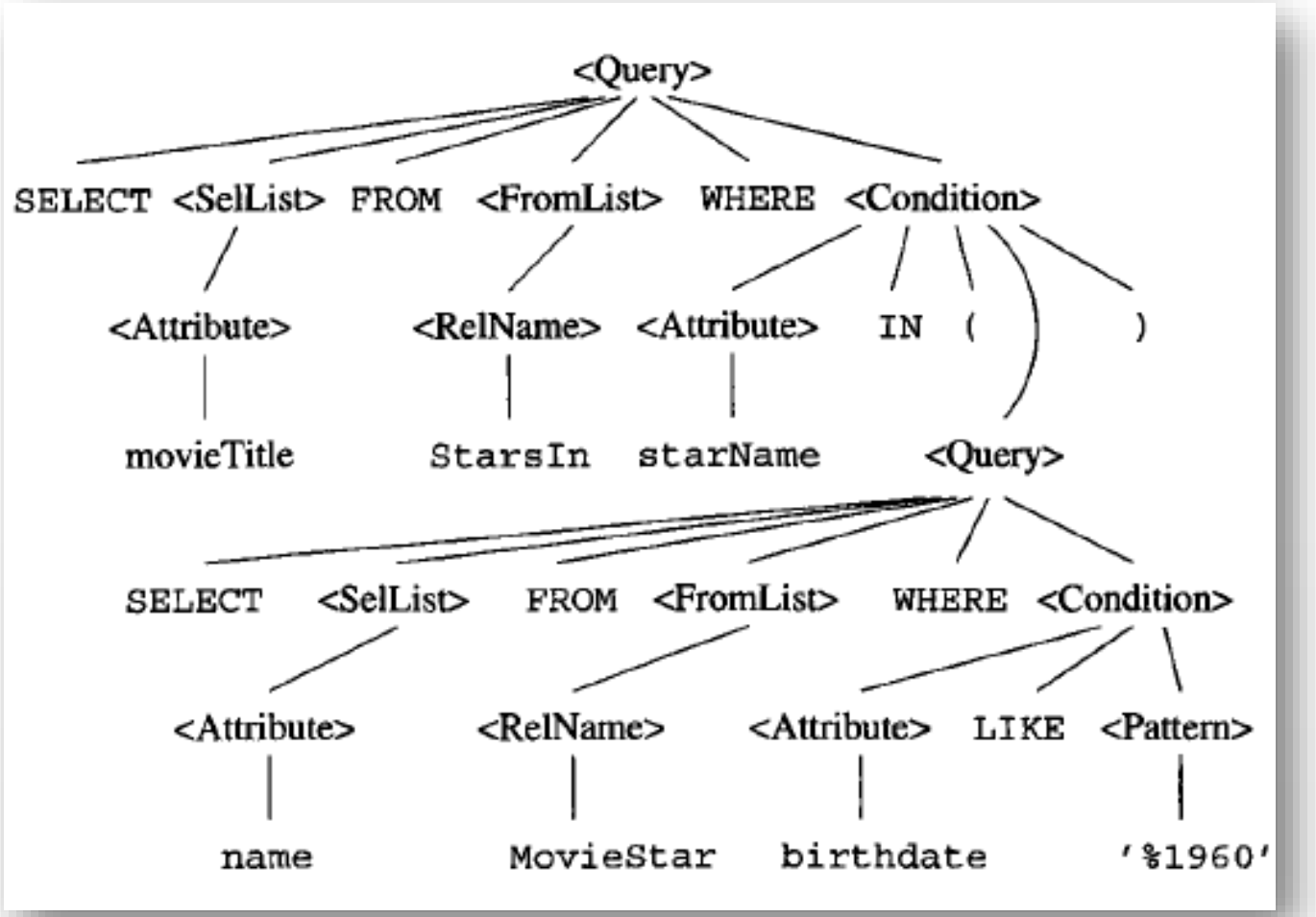
Procesiranje upita

Procesiranje koraci



Parsiranje

SELECT title
FROM StarsIn
WHERE starName IN (
    SELECT name
    FROM MovieStar
    WHERE birthdate LIKE ‘%1960’
)



Parsiranje

- Čvorovi stabla sadrže
  - Atome
    - Ključne reči
    - Imena atributa ili relacija
    - kontsante
    - Zagrade
    - Operatori algebarski, relacioni, logički
  - Sintaksičke kategorije
    - Nazivi familije delova upita

    <Query> - some queries in the common select-from-where form
    <Condition> - any expression that is a condition

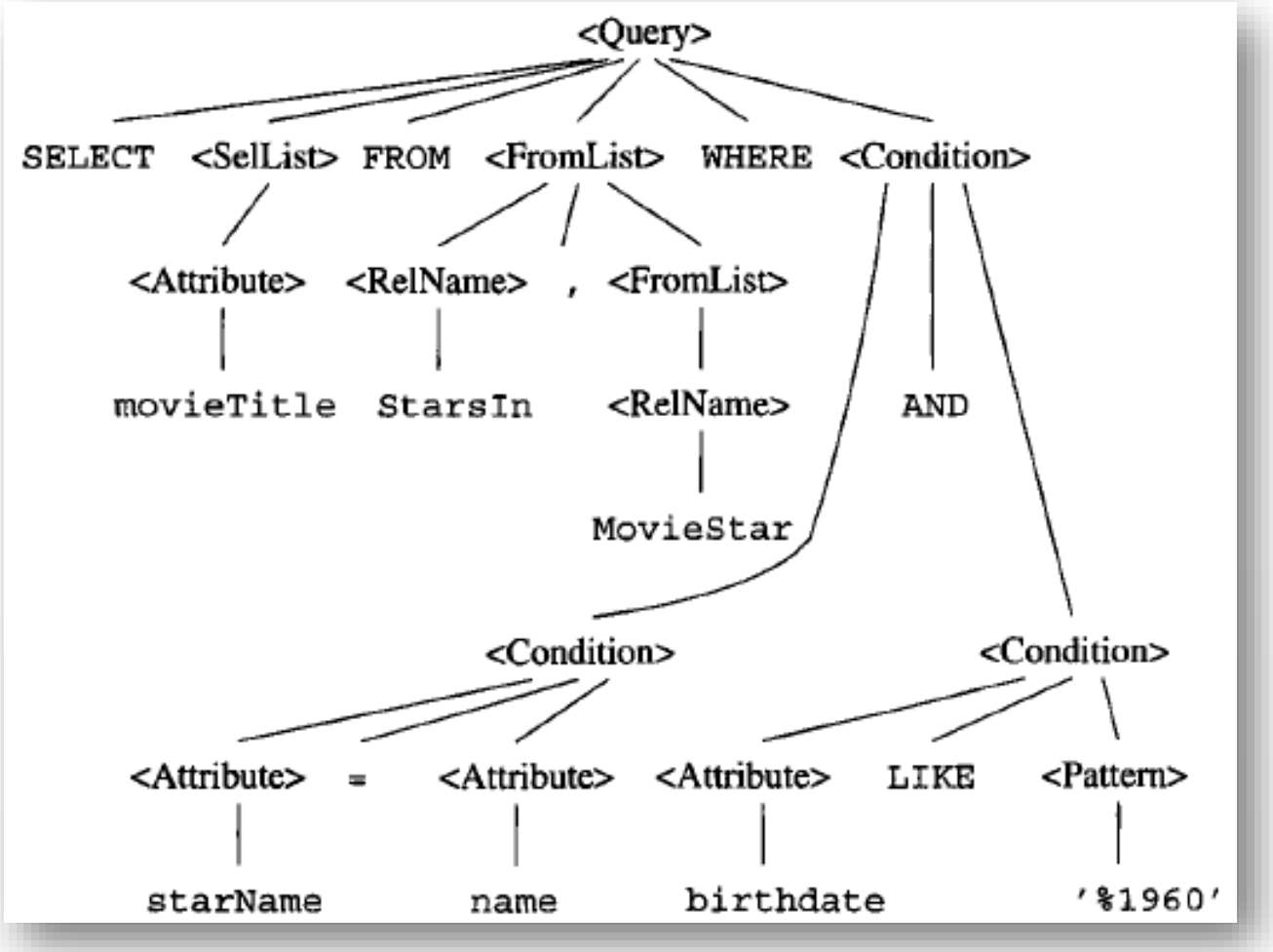
Parsiranje

SELECT title

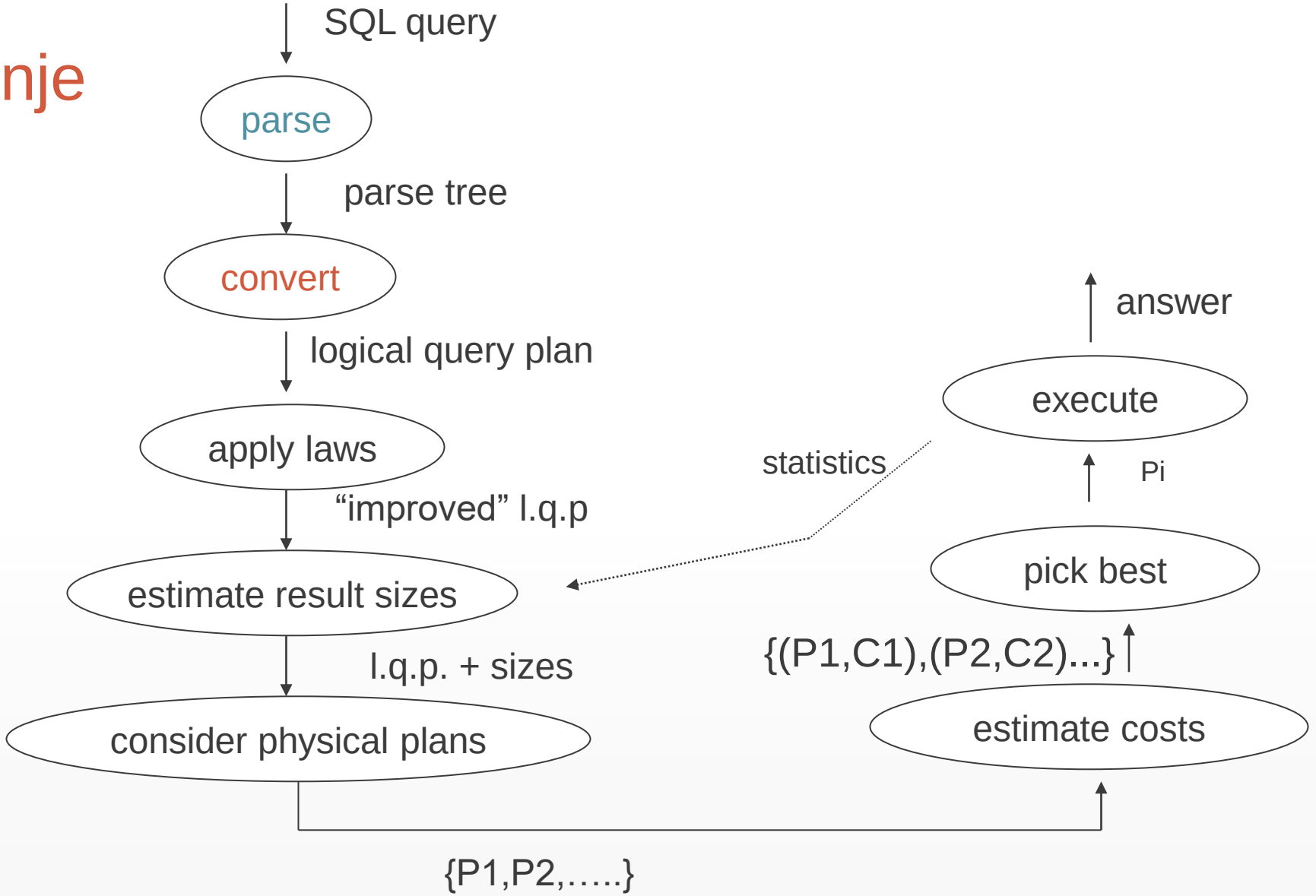
FROM StarsIn, MovieStar

WHERE starName = name AND

birthdate LIKE ‘%1960’



Procesiranje koraci



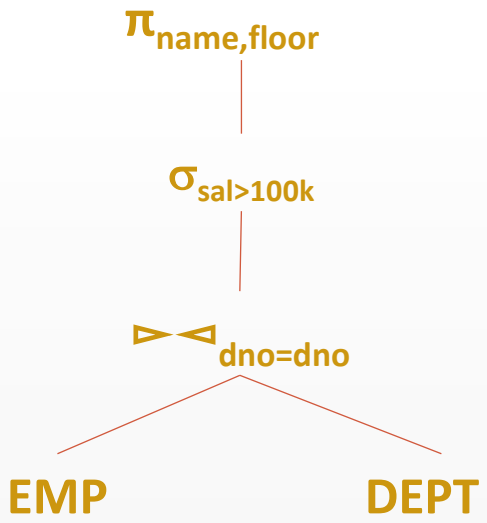
Preprocesiranje

- Nad parsnim stablom se vrši semantička provera:
  - Relacija
  - Atributa
  - Tipova
- Preprocesiranje referenci na poglede, tj. menjaju se – rezultujuće stablo je upit nad baznim tabelama.

Kreiranje logičkog plana (nakon preprocesiranja)

- Zamena čvorova i struktura iz parsnog drveta operatorima relacione algebre, tj.

  Kreiranje početnog logičkog plana izvršavanja.



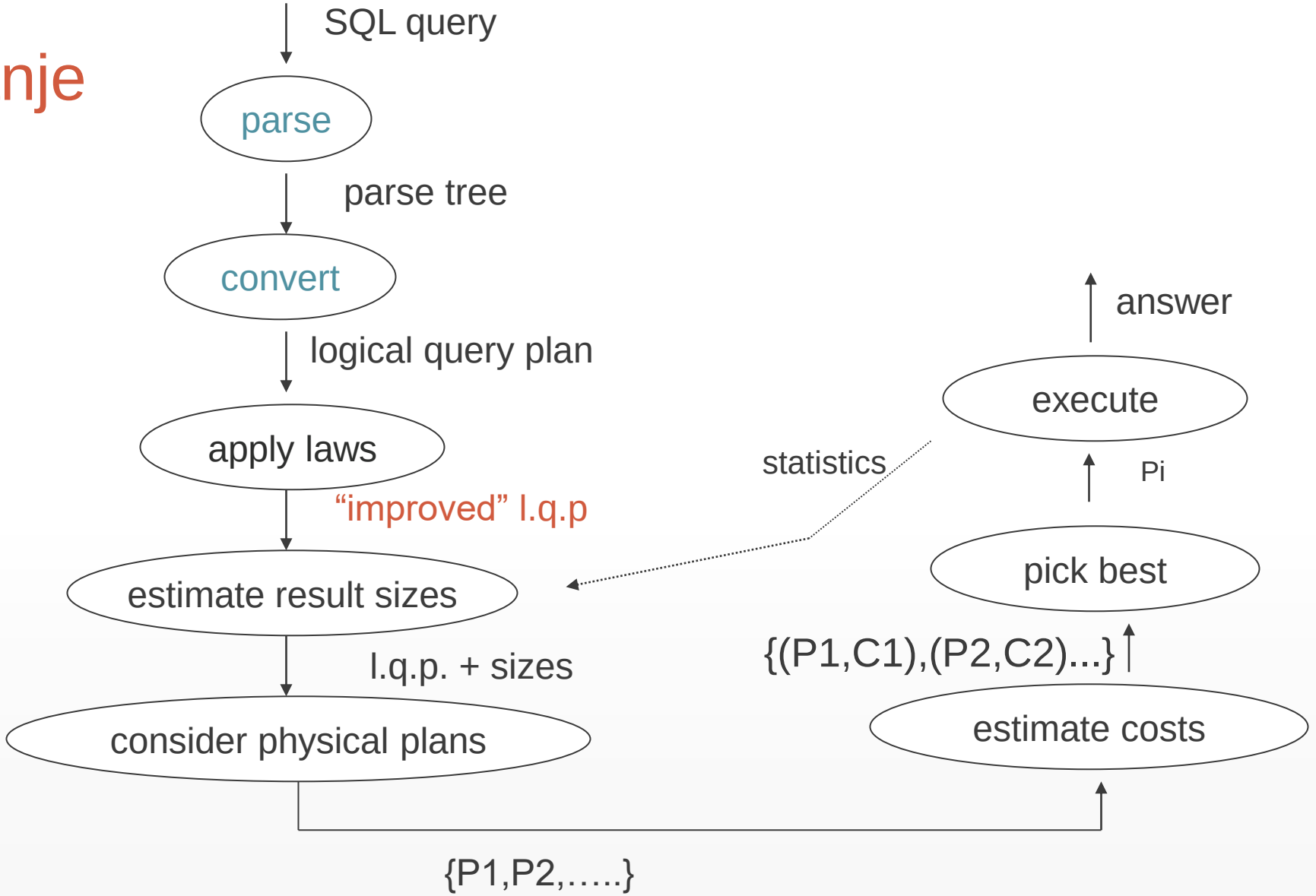
select name, floor
from emp join dept on emp.dno=dept.dno
where sal > 100K

emp(name, age, sal, dno)
dept(dno, dname, floor, mgr, ano)
act(ano, type, balance, bno)
bank(bno, bname, address)

Koraci – LP rewrite

Procesiranje upita

Procesiranje koraci



Ekvivalentna stabla

- Od početnog se kreiraju ekvivalentna stabla logičkog plana da bi se među njima odabrali oni za koje se očekuje da će zahtevati najkraće vreme izvršavanja.
- Primenom algebaskih zakona se dobijaju stabla koja rezultuju jednakim rezultatom za svaku legalnu intancu baze.

Komutativnost i asocijativnost

Algebarski zakoni

Važe za uniju, presek, ekvi-spajanje, dekartov proizvod (se tretira kao specijalan slučaj spajanja)

- R ⋈ S = S ⋈ R
- R ⋈ (S ⋈ T) = (R ⋈ S) ⋈ T

Kod teta spajanja asocijativnost ne vazi uvek

R(a,b), S(b,c), T(c,d)

$(R \bowtie_{R.b>S.b} S) \bowtie_{a<d} T$ <≠> $R \bowtie_{R.b>S.b} (S \bowtie_{a<d} T)$

Selekcija

Algebarski zakoni

$$\sigma_{C_1 \text{ AND } C_2}(R) = \sigma_{C_1}(\sigma_{C_2}(R)).$$

$$\sigma_{C_1 \text{ OR } C_2}(R) = (\sigma_{C_1}(R)) \cup_S (\sigma_{C_2}(R)).$$

$$\sigma_{C_1}(\sigma_{C_2}(R)) = \sigma_{C_2}(\sigma_{C_1}(R))$$

Pod uslovom da se unija tretira kao skupovna.

Selekcija (2)
Algebarski zakoni

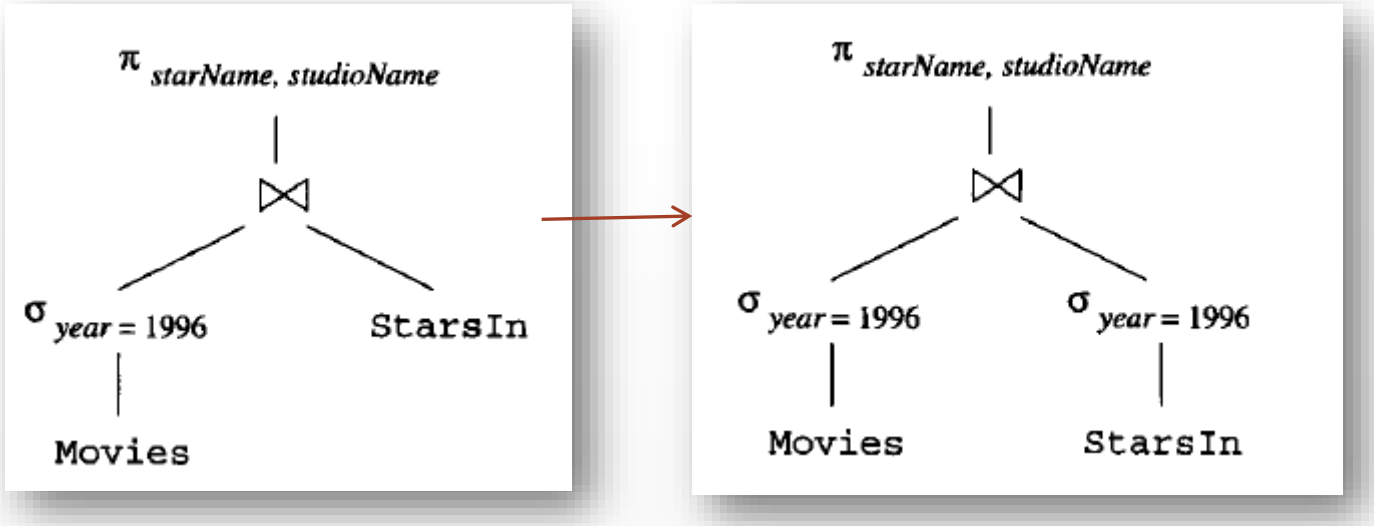
- Pomeranje selekcije niz stablo – korisna akcija
- Primena selekcije na binomne izraze koji sadrže:
  - Uniju – selekcija se mora primeniti na oba argumenta.
  - Razliku – selekcija se mora primniti na prvi argument razlike (umanjenik).
  - Ostali operatori (presek, unija, spajanje) – neophodna je primena na jedan operand. U slučaju spajanja i proizvoda
    - ne mora biti moguće primeniti selekciju na oba operanda,
    - ako je moguće primeniti na oba operanda može se, a ne mora dobiti ubrzanje.

Selekcija

- Pomeranje selekcije na gore
- Primena selekcije na sve moguće grane može da zahteva pomeranje selekcije na gore, a zatim spuštanje niz stablo.

```
CREATE VIEW Movies0fl996
AS
SELECT *
FROM Movies
WHERE year = 1996;

SELECT starName, studioName
FROM MoviesOf1996 NATURAL JOIN Starsln ;
```



Projekcija

Algebarski zakoni

- Spuštanje projekcije niz stablo zahteva uvođenje novih (drugačijih) projekcija.
- Osnovno pravilo – moguće je uvesti projekciju bilo gde u stablu pod uslovom da samo ako eliminiše atribute koje ni jedan operator iznad ne koristi.
- $\pi_L(R \bowtie S) = \pi_L(\pi_M(R) \bowtie \pi_N(S))$

  M i N su atributi koji učestvuju u spajanju, a pripadaju skupu atributa L i redom relaciji R, odnosno S.

Uz analogne uslove važi i

- $\pi_L(R \bowtie_C S) = \pi_L(\pi_M(R) \bowtie_C \pi_N(S))$
- $\pi_L(R \times S) = \pi_L(\pi_M(R) \times \pi_N(S))$

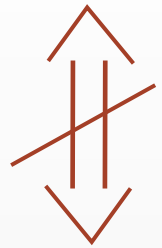
Projekcija (2)

Algebarski zakoni

- Ne mogu se spustiti niz skupovne verzije unije, preseka i razlike.

R(a,b) = {(1,2)}, S(a,b) = {(1,3)}

$$\pi_a(R \cap S) = \pi_a(\emptyset) = \emptyset$$



$$\pi_a(R) \cap \pi_a(S) = \{(1)\} \cap \{(1)\} = \{(1)\}$$

Spajanje i proizvod

Algebarski zakoni

- $R \bowtie_C S = \sigma_C(R \times S)$
- $R \bowtie S = \pi_L(\sigma_C(R \times S))$

C je uslov koji izjednačava sve zajedničke atribute

L lista koja uključuje pojedan primerak zajedničkih atributa i sve ostale

Uređivanje spajanja

Algebarski zakoni

- Dobar raspored operacija spajanja - važan za smanjenje veličine međurezultata

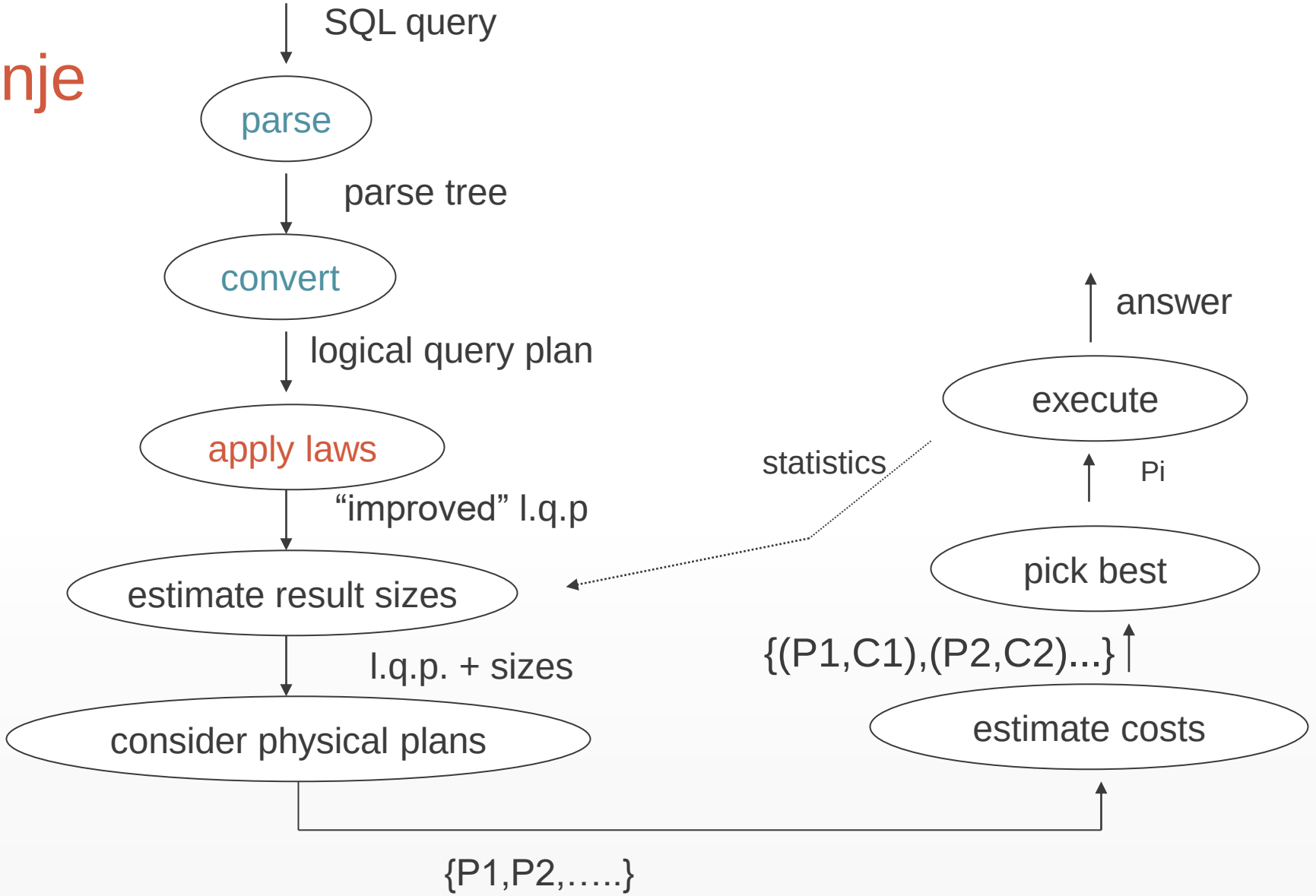
$$\pi_{\text{nazivpredmeta}} (\sigma_{\text{mesto='Kruševac'}}(\text{student}) \rhd\lhd \text{studira} \rhd\lhd \text{planpredmet})$$

$$\pi_{\text{nazivpredmeta}} (\text{studira} \rhd\lhd \text{planpredmet} \rhd\lhd \sigma_{\text{mesto='Kruševac'}}(\text{student}))$$

Koraci – LP heuristike

Procesiranje upita

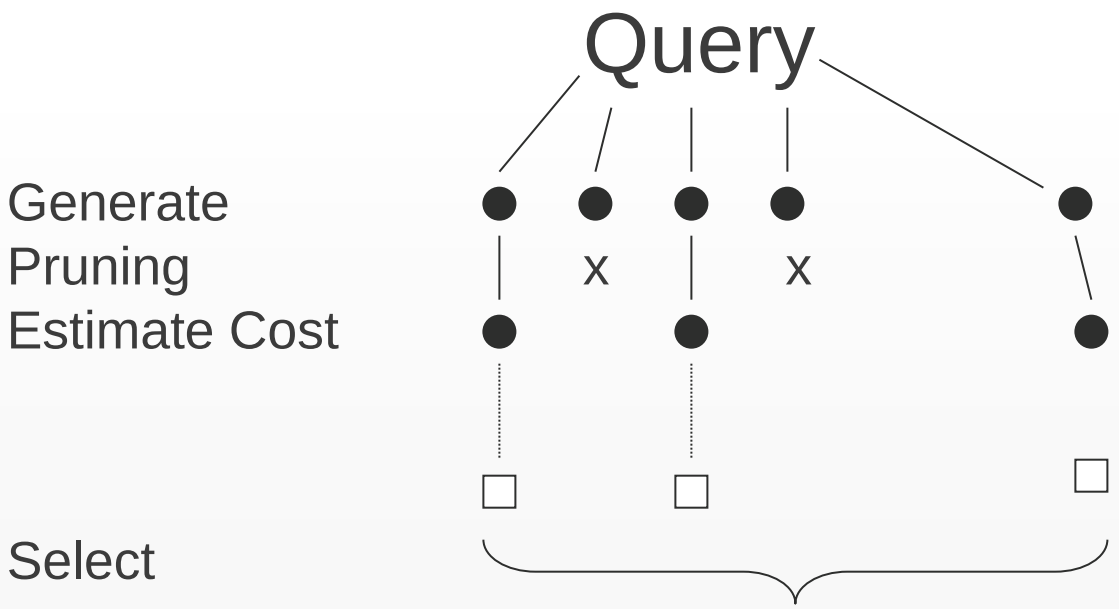
Procesiranje koraci



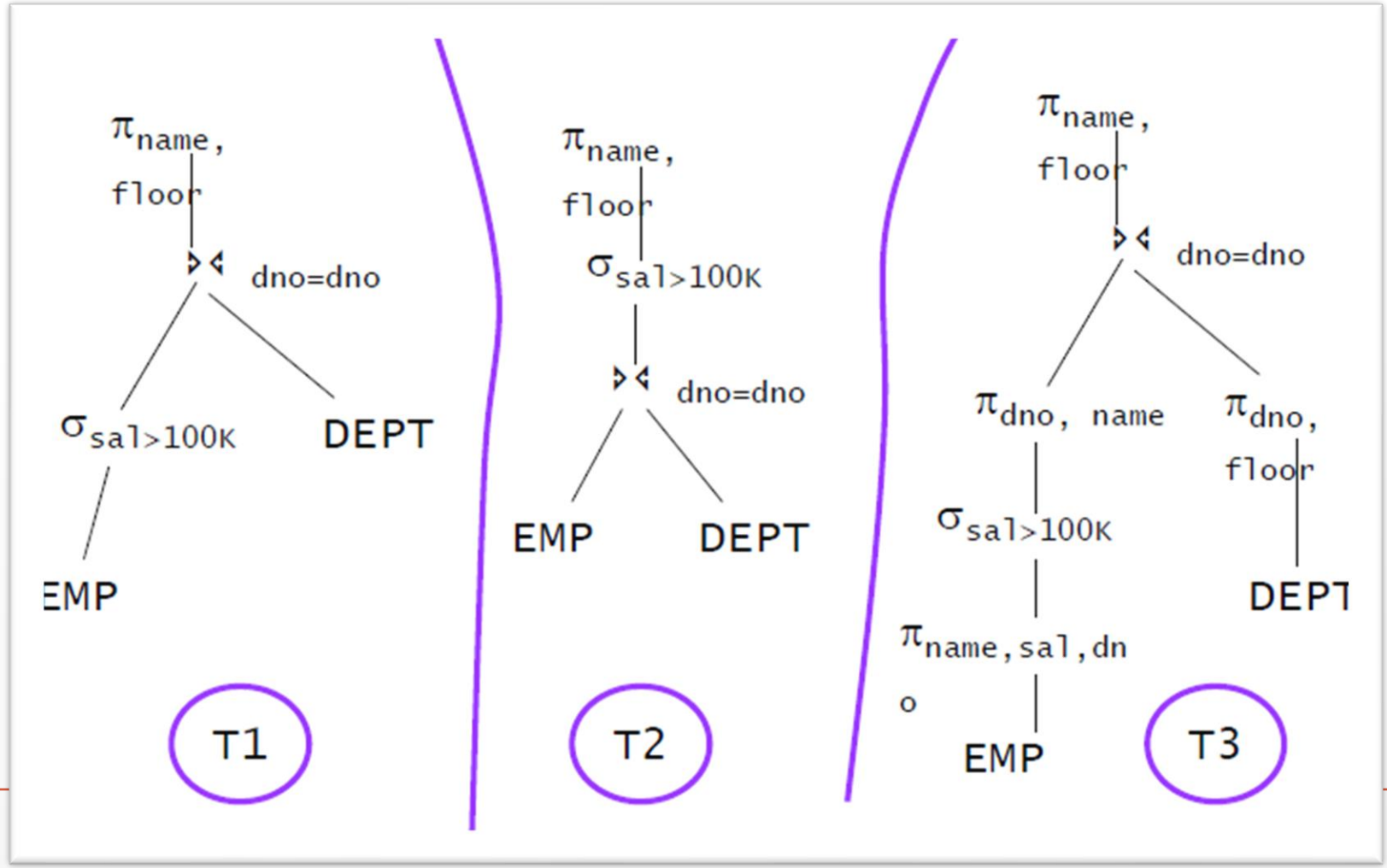
Restrikcije algebarskog prostora

- Veliki algebarski prostor se sužava primenom odgovarajućih heuristika
- **Restrikcija 1**

Dozvoliti samo stabla u kojima su selekcija i projekcija procesirani najranije moguće



Restrikcije algebarskog prostora (2)



Procesiranje selekcije i projekcije u fizičkom planu

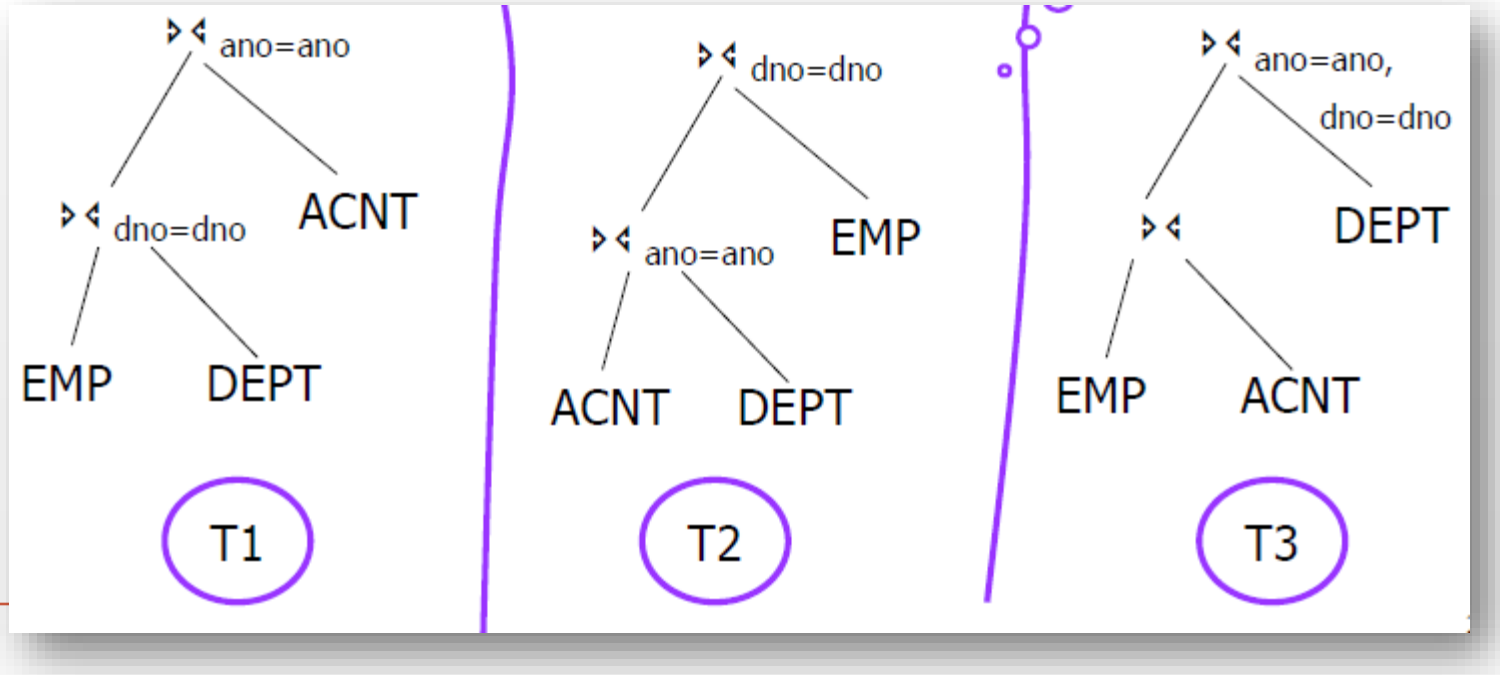
- Selekcija i projekcija se izvršavaju “On the fly”
- Ne zahtevaju pisanje po disku
- Selekcija se izvodi pri prvom čitanju relacija
- Projekcija se izvodi tokom određivanja rezultata prethodne operacije/akcije

Restrikcije algebarskog prostora (3)

- **Restrikcija 2**

Proizvod relacija se ne izvodi, osim u slučaju da je nemoguće izbeći ga

```
select name, floor, balance
from emp, dept, acnt
where emp.dno=dept.dno and
        dept ano = acnt ano
```



Restrikcije algebarskog prostora (4)

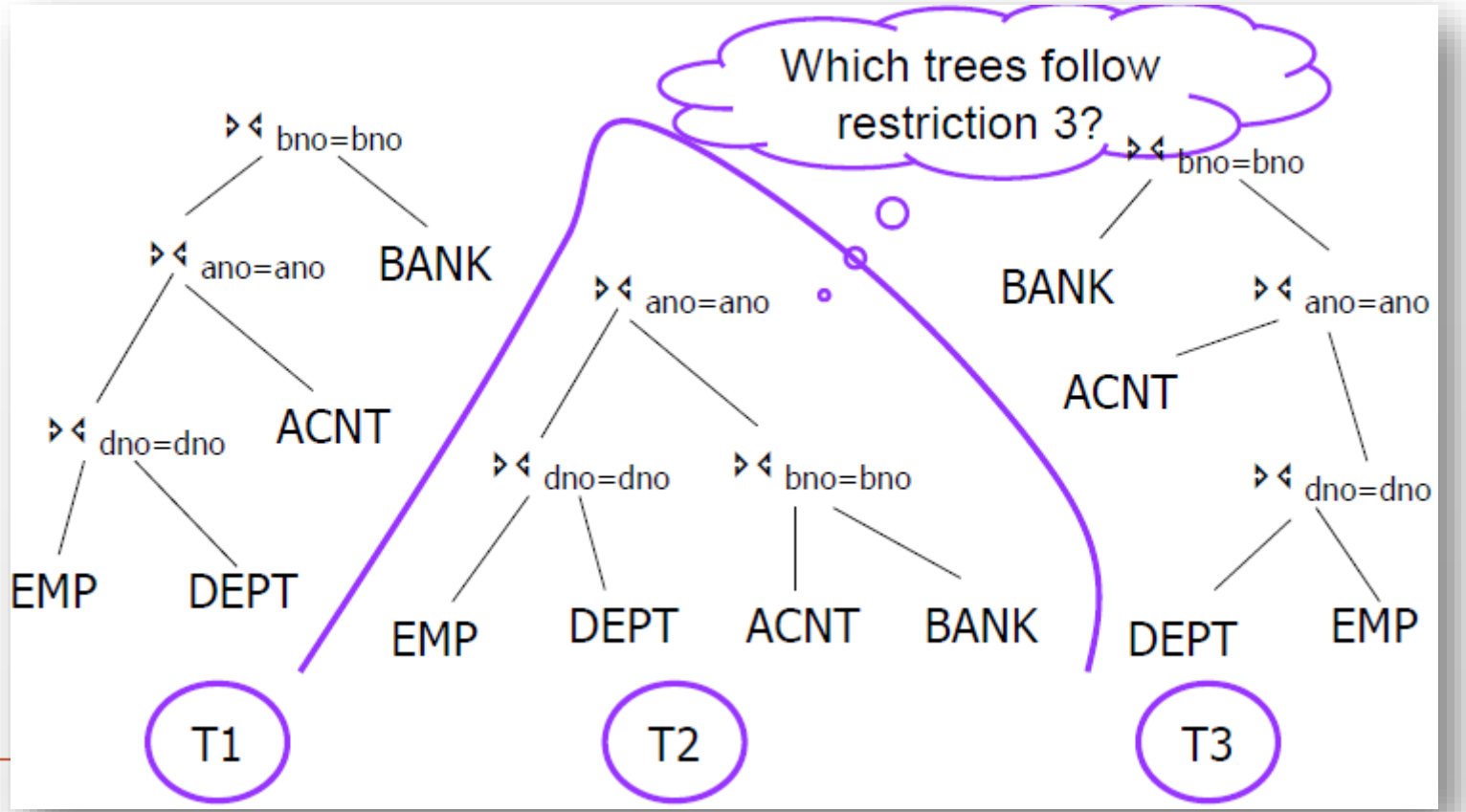
- **Restrikcija 3**

Unutrašnji operand spajanja je bazna relacija, ne međurezultat (left-deep plan)

Levi operand spajanja se naziva **spoljašnjim**, a desni **unutrašnjim**.

Restrikcije algebarskog prostora (5)

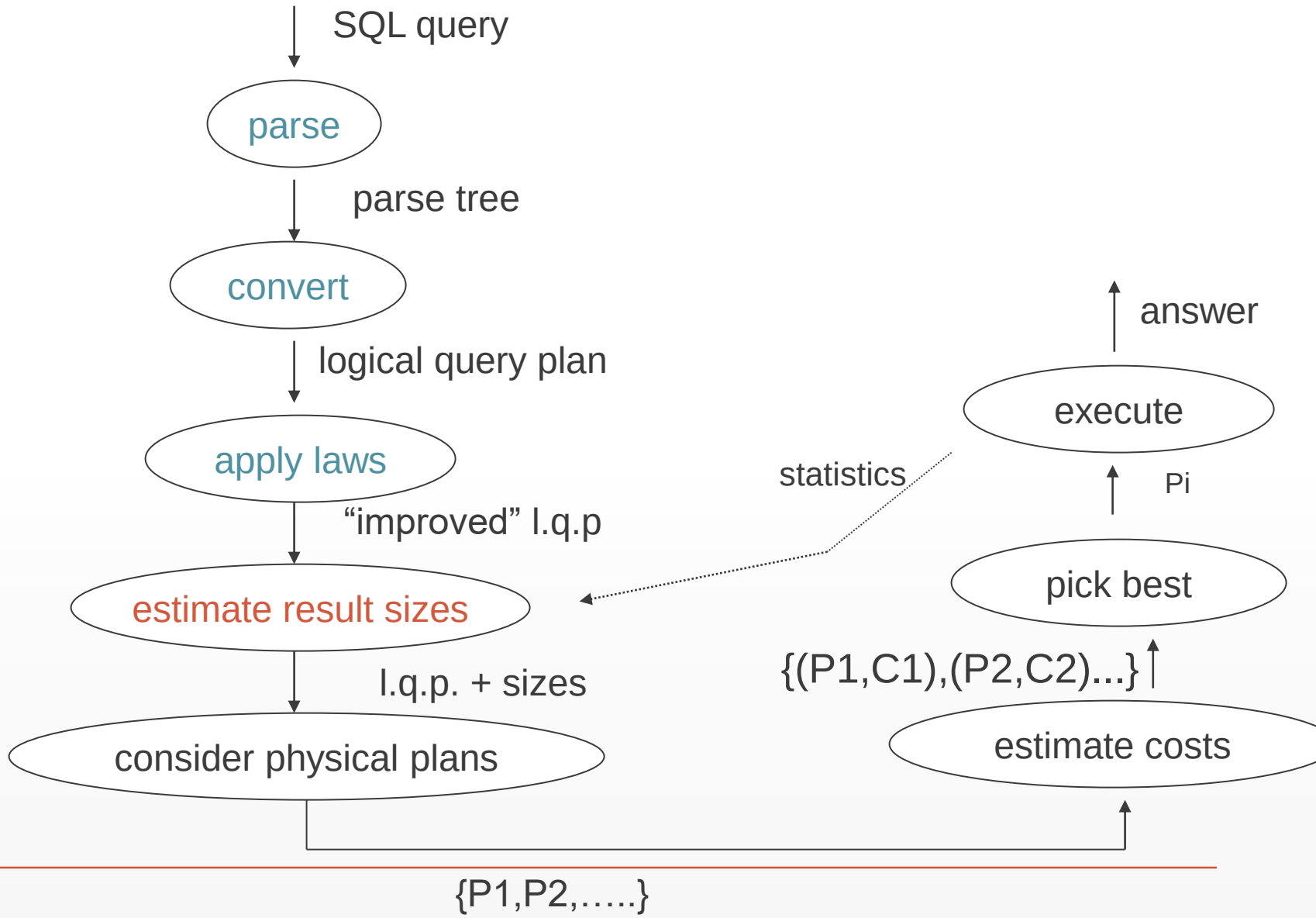
```
select name, floor, balance
from emp, dept, acnt, bank
where emp.dno=dept.dno and
      dept.ano=acnt.ano and
      acnt.bno = bank bno
```



Procena veličine međurezultata

Ocena logičkog plana

Procesiranje koraci



Procena veličine međurezultata

- Logički plan -> više fizičkih planova
- Za odabir fizičkog plana vrši se procena kompleksnosti svakog plana pojedinačno.
- Da bi se odredila kompleksnost potrebna je procena veličine međurezultata.
- Za procenu veličine međurezultata se koriste pravila koja:
  - Daju dovoljno dobre procene
  - Su jednostavna za izvršavanje.
  - Su logički konzistentna
    procenjena veličina ne treba da zavisi od načina na koji je relacija dobijena

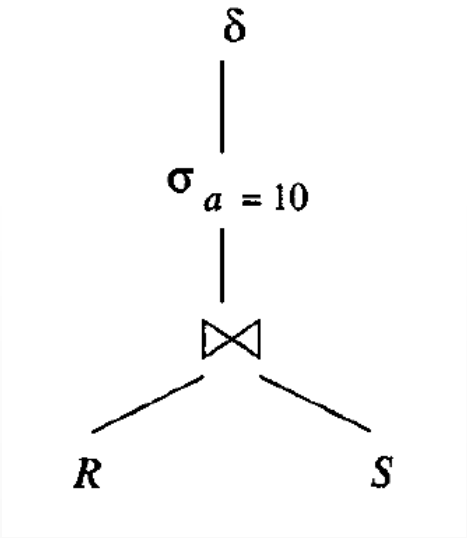
Statistički podaci

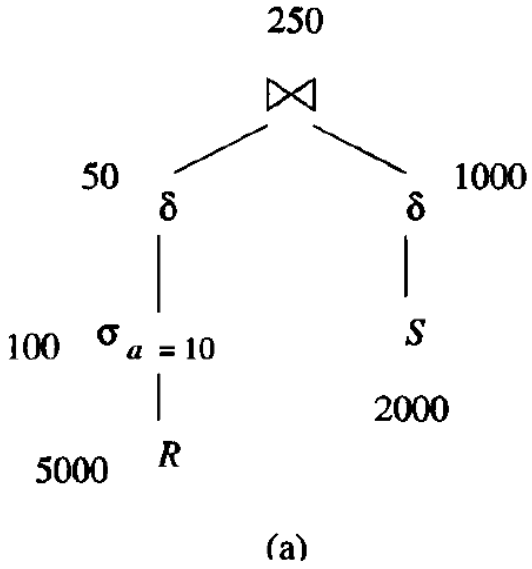
- Sistemski katalog sadrži statističke podatke koji se koriste pri proceni veličine međurezultata, kao što su:
  - Broj torki u relaciji
  - Broj blokova koji sadrže torke relacije
  - Veličina torki relacije u bajtovima
  - Broj torki relacije koje se mogu smestiti u jedan blok (blocking factor)
  - Broj različitih vrednosti pojedinih atributa relacije (u tekućoj instanci baze).
- Uvedimo oznake:

B(R) – broj blokova potrebnih za relaciju **R**,
T(R) – broj torki relacije **R**,
V(R, a) – broj različitih vrednosti atributa **a** relacije **R**.

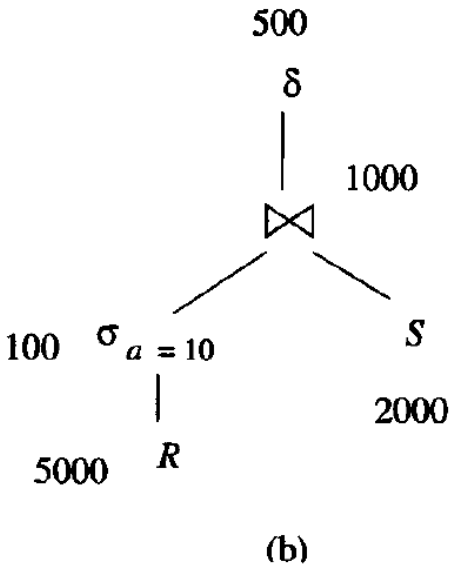
Primer procene

$R(a,b)$	$S(b,c)$
$T(R) = 5000$	$T(S) = 2000$
$V(R,a) = 50$	
$V(R,b) = 100$	$V(S,b) = 200$
	$V(S,c) = 100$





(a)



(b)

Procena veličine projekcije

- U većini slučajeva projekcija smanjuje količinu podataka koji su ‘stigli do nje’.

R(a,b,c)

a,b - integers of 4 bytes each
c - a string of 100 bytes

tuple headers - 12 bytes
block - 1024,
block header - 24 bytes

Then each tuple of R requires 120 bytes. We can thus fit 8 tuples in one block. For T(R) = 10,000 -> B(R) = 1250.

S= $\pi_{a,b}$(R) - tuple - 20 bytes, B(S) = 200

Procena veličine selekcije

- **$S = \sigma_{A=c}(R)$**

$$T(S) = T(R) / V(R,A)$$

- **$S = \sigma_{A<c}(R)$**

$$T(S) = T(R)/2,\ T(S) = T(R)/3$$

- **$S = \sigma_{A\neq c}(R)$**

$$T(S) = T(R)*(1-V(R,A))/V(R,A))$$

Procena veličine selekcija sa složenim predikatima

- **$S = \sigma_{\theta 1 \text{ AND } \theta 2 \text{ AND } \ldots \text{ AND } \theta n}(R)$**

  Broj torki polazne relacije pomnožen faktorom selektivnosti za svaki term u predikatu

  $$T(S) = s_1 * s_2 * \ldots * s_n * T(R)$$

- **$S = \sigma_{\theta 1 \text{ OR } \theta 2}(R)$**

  Ako R sadrži n torki, od kojih m_i 'zadovoljava' C_i

  $$T(S) = (1 - (1 - m_1/n)(1 - m_2/n)) * T(R)$$

Procena veličine spajanja

- **R ⋈ S (θ-join)**
  - Može se tretirati kao selekcija za kojom sledi proizvod
  - Ako je R∩ S = ∅, tada je broj n-torki jednak broju n-torki u R x S
  - Ako je R∩ S ključ relacije R tada broj torki u rezultatu neće biti veći od broja torki u S.
  - Ako je je R∩ S strani ključ u S koji se referencira na vrednosti primarnog ključa u R tada je broj torki u rezultatu jednak broju torki u S.
  - Ako je R∩ S = {A}, gde a nije ključ tada je
    pod pretpostavkom da se sve vrednosti atributa A koje se javljaju u jednoj relaciji pojavljuju i u drugoj, tj.

    V (R ⋈ S, A) = V(R, A)

    broj torki u rezultatu je

    $$T(R \bowtie S)=T(R)T(S)/\max(V(R,Y),V(S,Y))$$

Još malo o statistici u katalogu

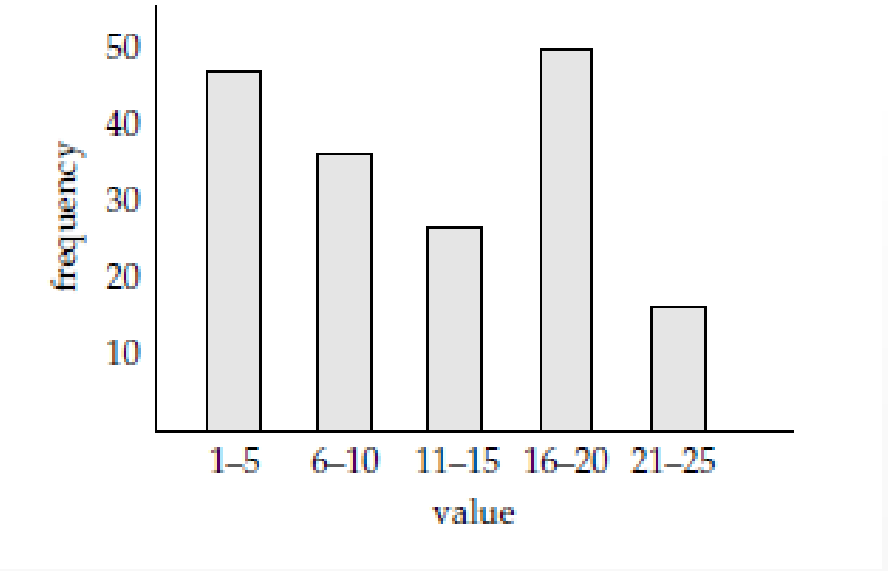
- Statistički podaci o relacija se periodično sračunavaju
  - ‘Neprecizna’ statistika je još uvek korisna sve dok se dosledno koristi u svim planovima koji se ocenjuju.
- Osvežavanje statitike može biti trigerovano automatski ili po zahtevu administratora.
- Računanje statistike nad celim relacijama može biti veoma skupo, naročito u računanju V(R,a). Zato se na osnovu stanja u uzorku vrši procena stanja u celoj tabeli.

Histogrami

- DBMS-ovi često koriste histograme kojima beleže rasporede vrednosti pojedinih atributa.
- Dobro za bolju procenu cene operacije spajanja.

- Equal-width.
  $V_0 \leq v < V_0+w$, $V_0+w < v < V_0+2w$, ...
- Equal-height.
- Most-frequent-values.

  1: 200, 0: 150, 5: 100, others: 550



Algoritmi operatora

Implementacija operatora

Tehnike koje se koriste pri implementaciji operatora su:

- **Indeksiranje** – na osnovu ideksa se vrsi odvajanje onih torki koje ce biti ispitivane pri realizaciji operacije
- **Iterisanje** – pretraga svih torki u tabeli, ili celog indeksa ukoliko sadrzi sve atribute koji se ispituju
- **Particionisanje** – particionisanjem torki prema kljucu se operacija razbija na kolekciju operacija nad manjim skupom podataka, uobicajene tehnike koje se koriste pri particionisanju su **sortiranje** i **hesiranje**

Pristup podacima (access paths)

Trazenim torkama u fajlu/tabeli je moguce pristupiti

- Sekvencijalnim citanjem fajla
- Pretragom indeksa + dobavljanjem odgovarajucih torki

Da bi bio upotrebljen za pristup torkama indeks mora da odgovara uslovu selekcije:

- **Heš indeks** odgovara uslovu (u KNF) ako postoji term oblika **atribut = vrednost** za svaki atribut u kljucu pretrage samog indeksa.
- **Uredjeni indeks** odgovara uslovu ako postoji term oblika **atribut operacija vrednost** za svaki atribut nekog prefiksa kljuca indeksa (prefiksi kljuca (a,b,c) su (a) I (a,b))

Pristup podacima (access paths) 2

- **Selektivnost** pristupa - broj strana/blokova (indeksnih + blokova sa podacima) koji dobavljaju za pretragu.
- Zastupljenost torki koje zadovoljavaju uslov (konjugat) predstavlja **faktor redukcije** traženog uslova.

ıash index H on Sailors with search key $\langle rname, bid, sid \rangle$

$rname = \text{'}Joe\text{'} \wedge bid = 5 \wedge sid = 3.$

$Npages(Sailors) \cdot \frac{1}{NKeys(H)}.$

Algoritmi implementacije operacija - Selekcija

$$\sigma_{R.attr\ \mathbf{op}\ value}(R)$$

- Ukoliko nema indeksa nad R.attr skenira se cela R
- Ukoliko postoji indeks u zavisnosti od uslova i toga da li je indeks klasterovan moguce je:
  - Suziti pretragu na manji broj strana
  - Dobiti direktne reference na torke koje se nalaze u uzastopnim stranama
  - Dobiti direktne reference u neklasterisanom indeksu (ako je broj torki koje treba dobaviti veci od 5%, tada je obicna sekvencijala pretraga bolji izbor od pretrage po kljucu)

Algoritmi implementacije operacija - Projekcija

- Projekcija bez eliminacije duplikata – jednostavno sekvencijalno učitavanje fajla sa podacima ili indeksa ako sadrži sve potrebne atribute
- Projekcija sa eliminacijom – primenjuje se neka od tehnika particionisanja

Primer:

- <sid,bid> projekcija nad Reserves
- Particionisanje –
  - Skeniranje Reserves za dobijanje parova <sid,bid>
  - Sortiranje dobijenih parova
- Po sortiranju se vrsi eliminacija duplikata

Algoritmi implementacije operacija - Spajanje

Posebna prezentacija

Sortiranje - External Sort-Merge

- Najčešće veličina fajla ne dozvoljava njegovo učitavanje u celosti i sortiranje na licu mesta i odmah, već se vrši u nekoliko koraka sa beleženjem sortiranih međurezultata.
- Osnovna ideja – podeliti na delove, delove sortirati, sortirane spajati u veće sortirane delove, sve dok rezultat spajanja ne bude cela tebela.
- Run (prolaz) – sortirani međurezultat

External Sort-Merge

M+1 – number of pages that can be placed in memory at the same moment.

1. **Create sorted runs**.

2. **Merge the runs (N-way merge)**. We assume that $N < M$.

    1. Use *N* blocks of memory to buffer input runs, and 1 block to buffer output. Read the first block of each run into its buffer page

    2. **repeat**

        1. Select the first record (in sort order) among all buffer pages

        2. Write the record to the output buffer. If the output buffer is full write it to disk.

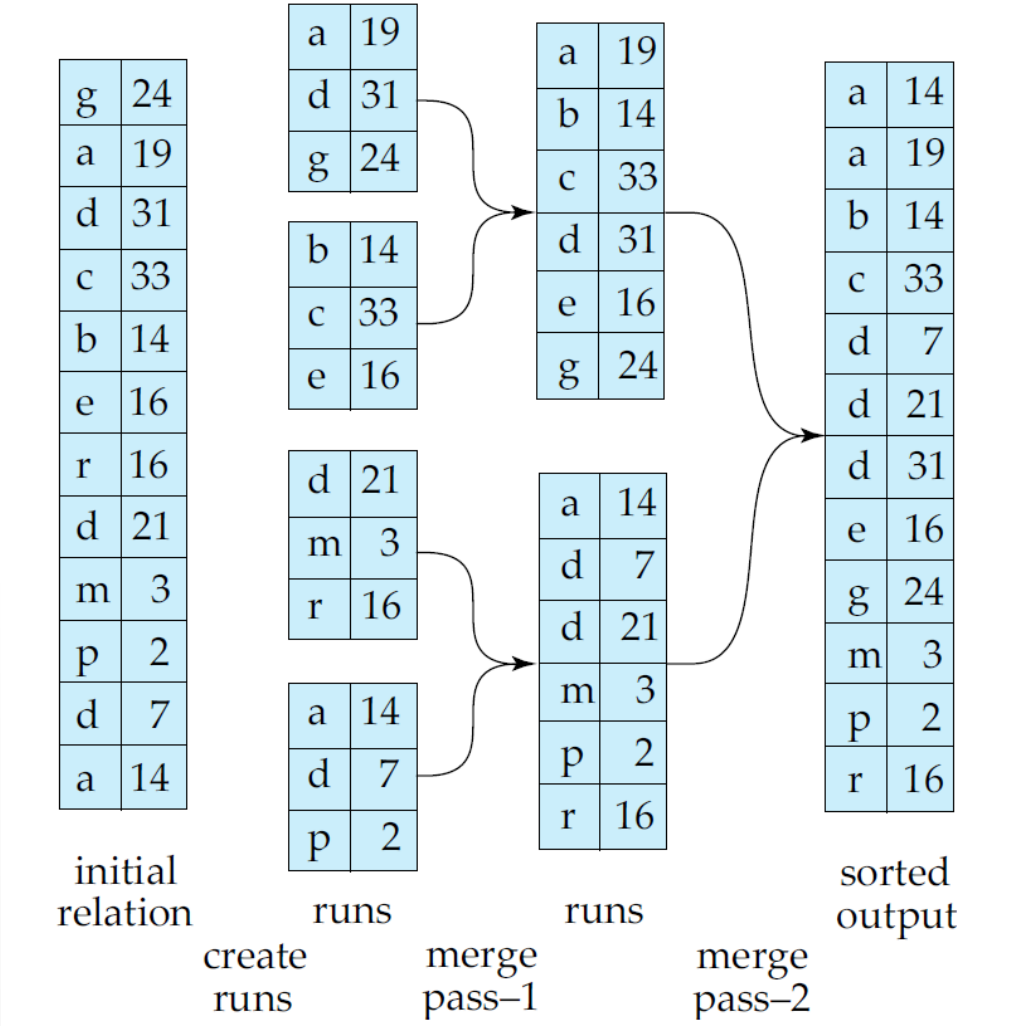
        3. Delete the record from its input buffer page.
           If the buffer page becomes empty **then**
           read the next block (if any) of the run into the buffer.

    3. **until** all input buffer pages are empty.

External Sort-Merge (2)

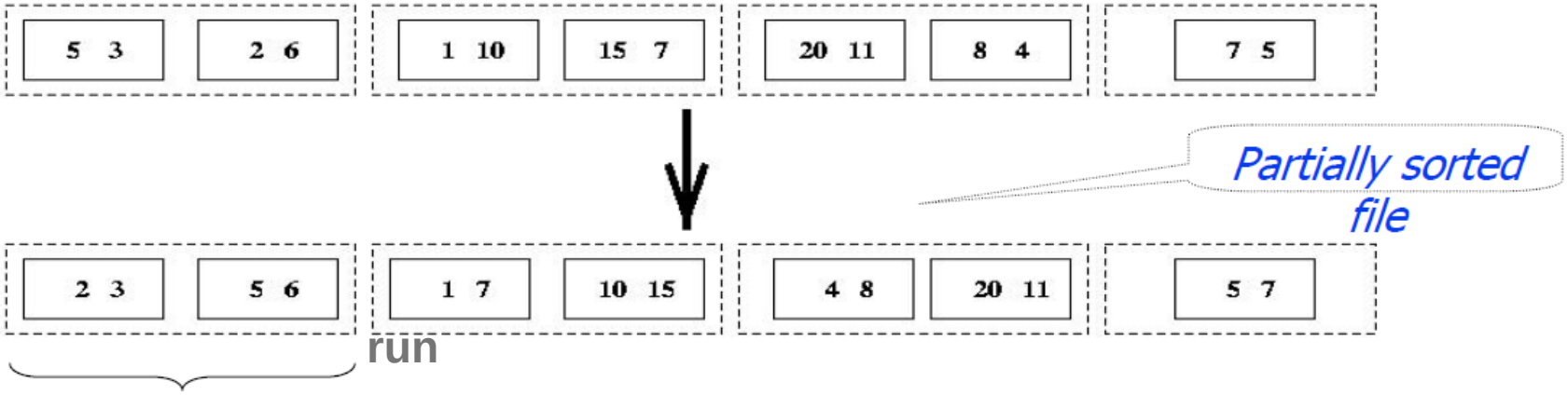
- If $N \geq M$, several merge *passes* are required.
  - In each pass, contiguous groups of M - 1 runs are merged.
  - A pass reduces the number of runs by a factor of M -1, and creates runs longer by the same factor.

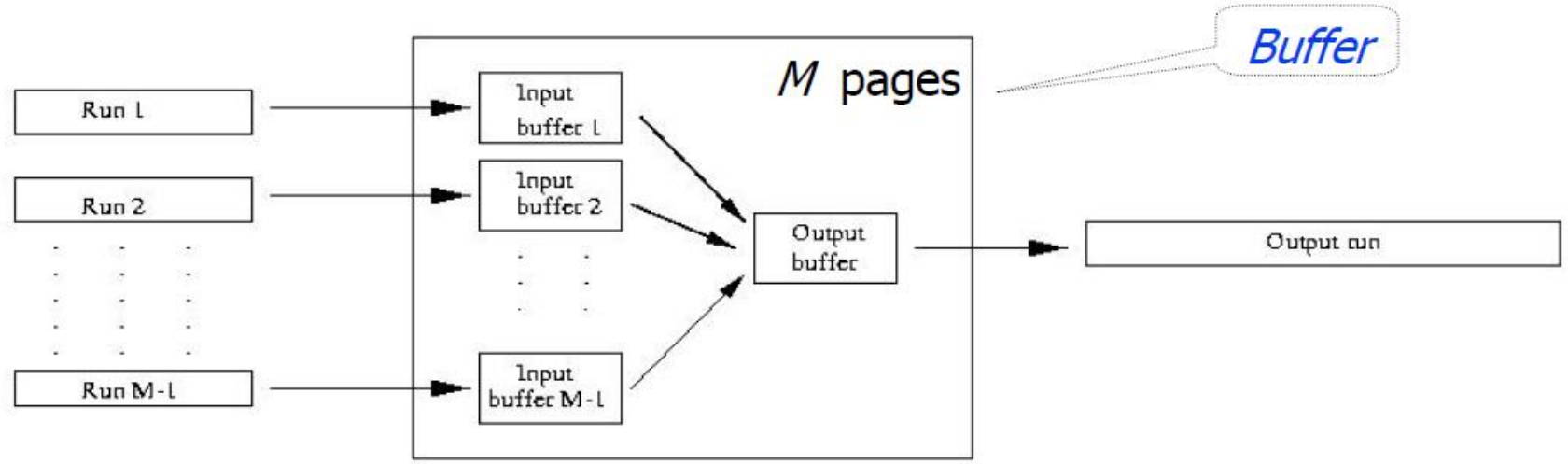
    E.g. If M=11, and there are 90 runs, one pass reduces the number of runs to 9, each 10 times the size of the initial runs
  - Repeated passes are performed till all runs have been merged into one.



External Sort-Merge (3)

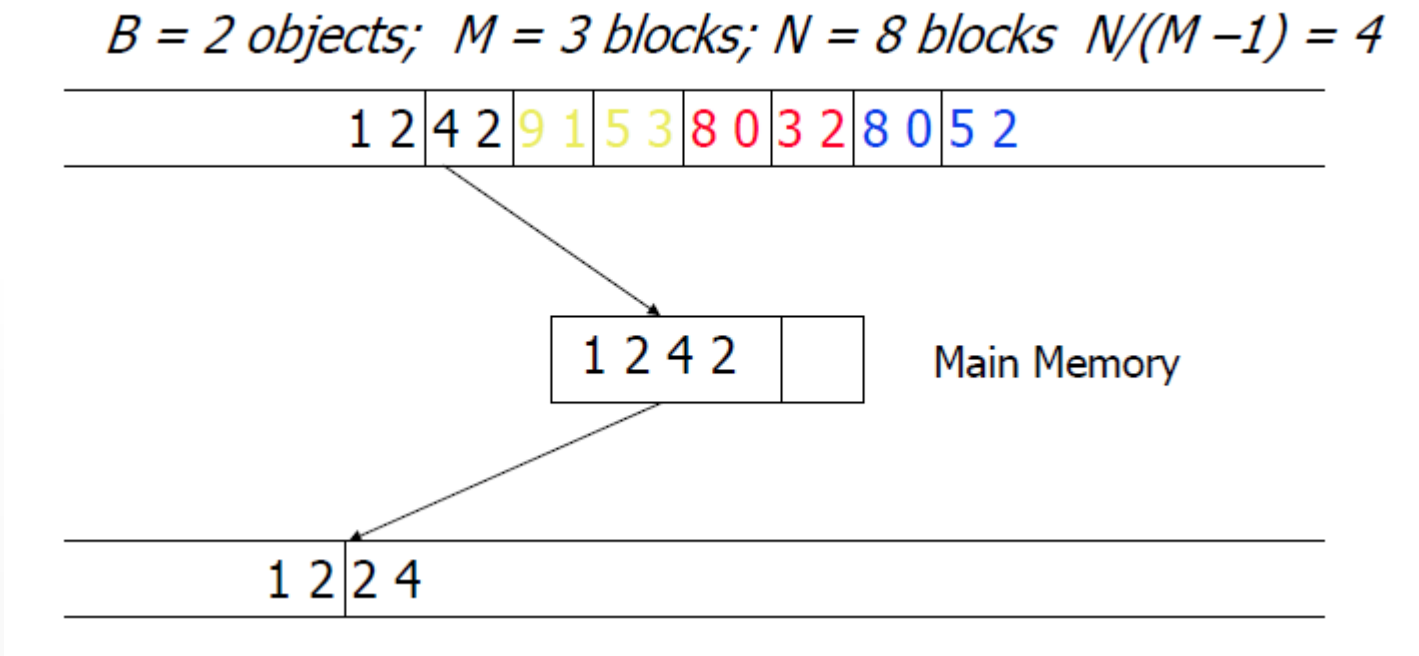
Partial sort phase: sort *M pages at a time; create N/M sorted runs on* mass store.
Merge Phase: *merge all runs into a single run using M-1 buffers for* input and 1 output buffer





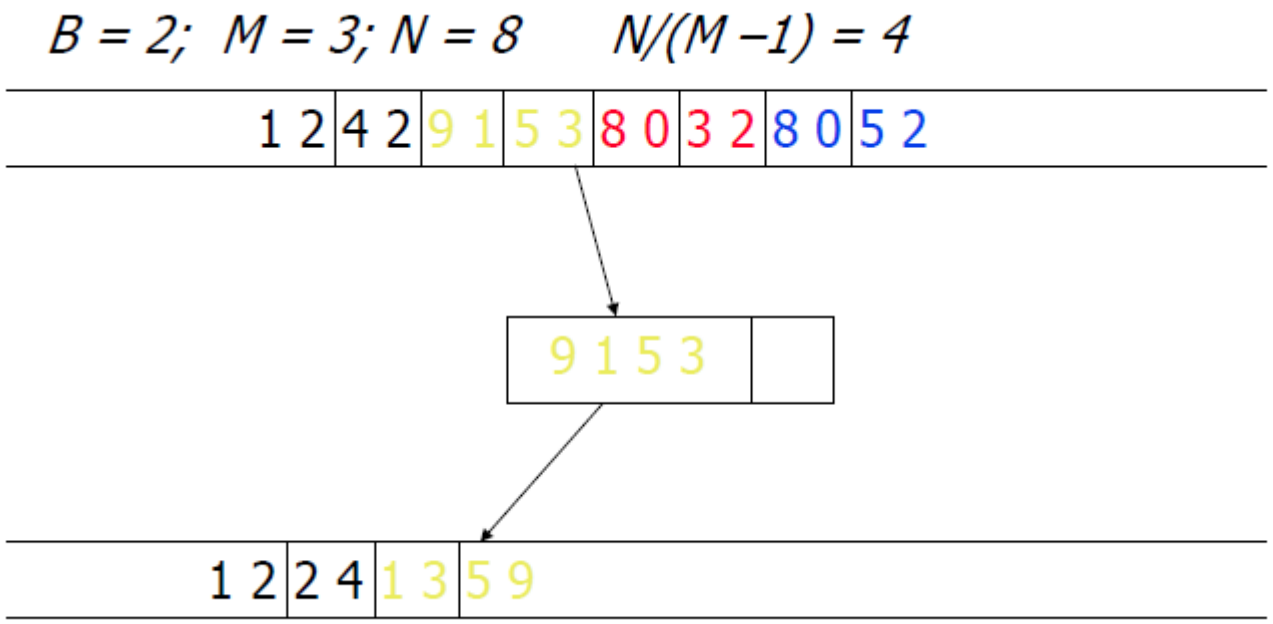
External Sort-Merge - primer

Phase 1: Create N/(M –1) sorted groups of size (M – 1) blocks each



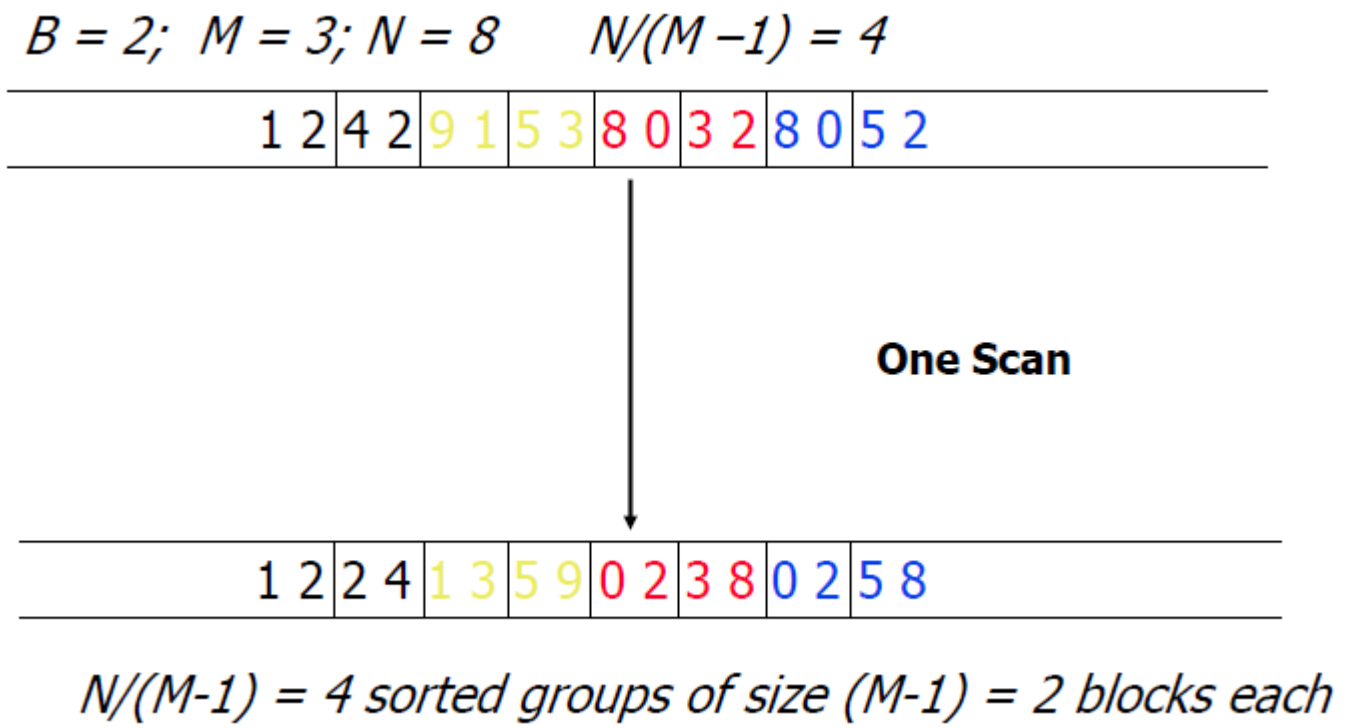
External Sort-Merge - primer

Phase 1



External Sort-Merge - primer

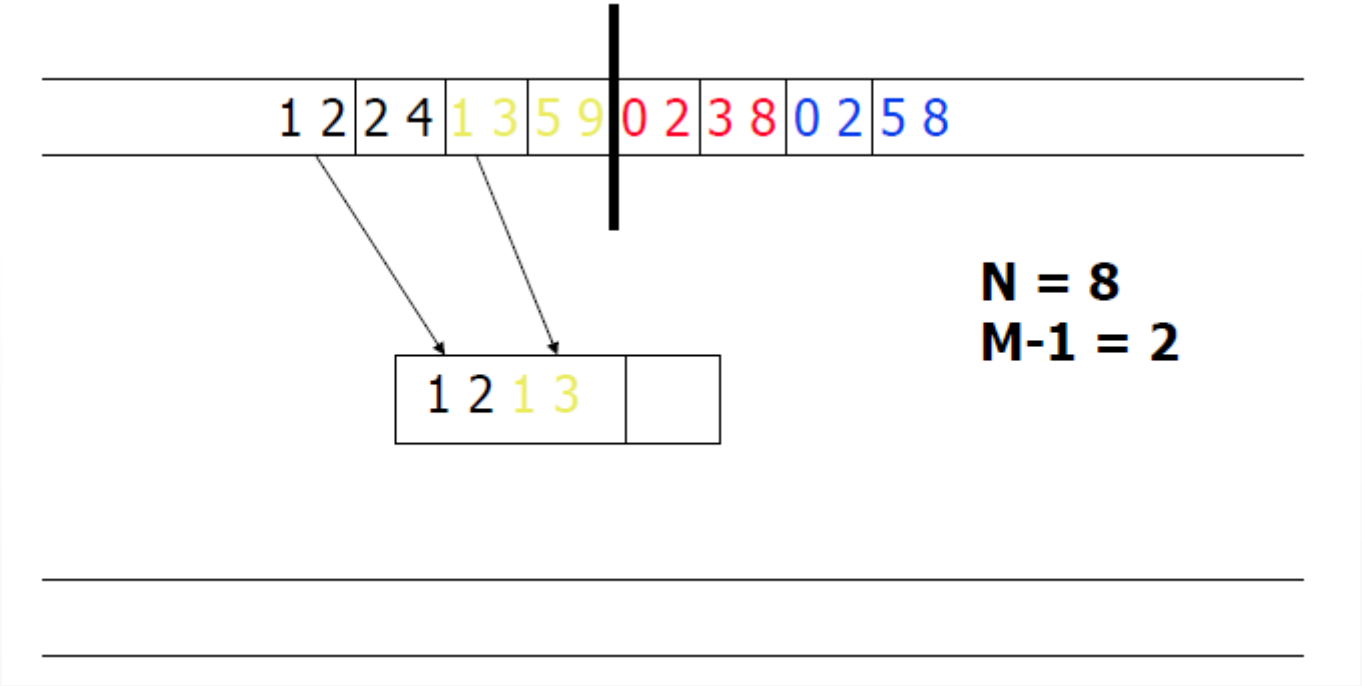
End of Phase 1



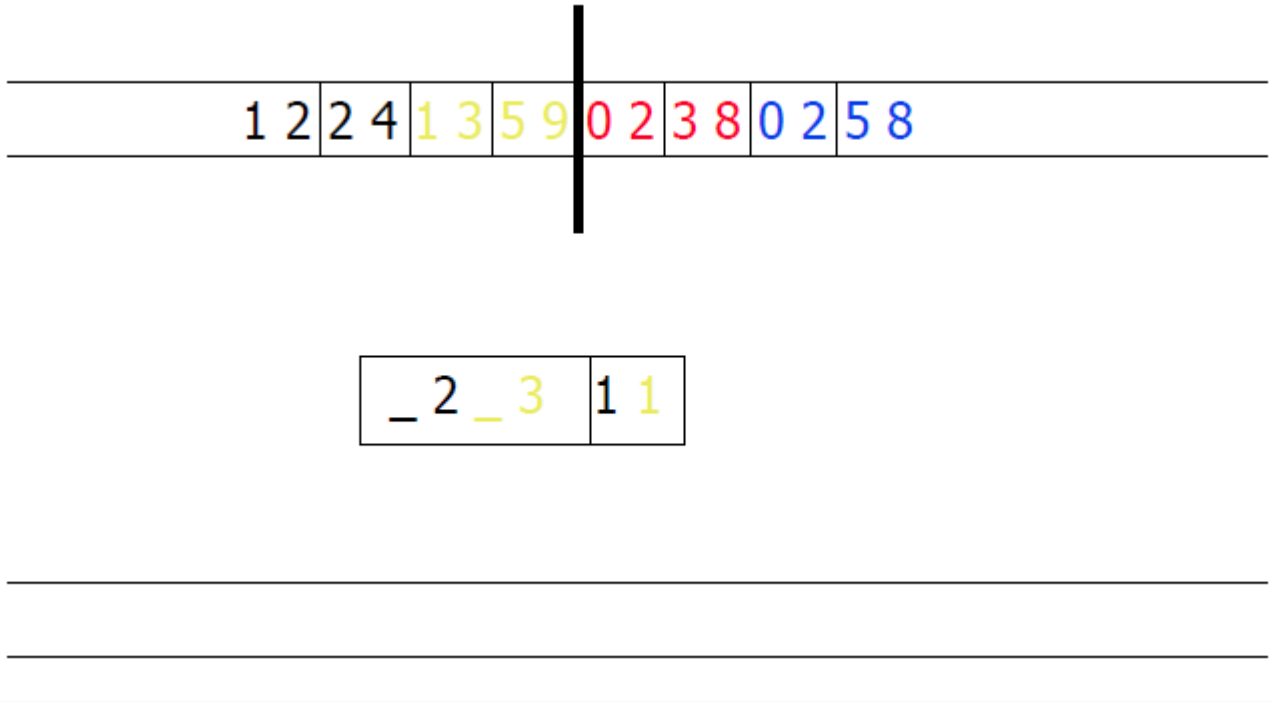
External Sort-Merge - primer

Phase 2 Merging Runs First Pass

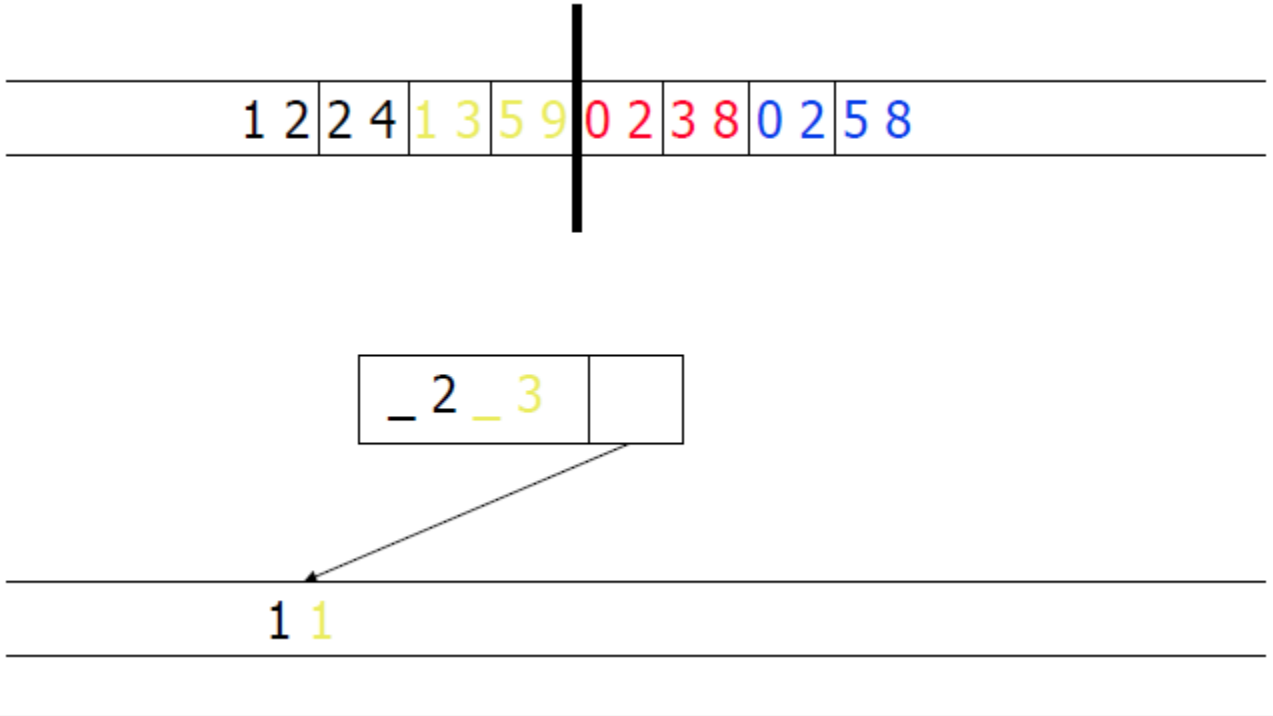
- Merge *(M-1) sorted groups into one sorted group*
- Iterate until all groups merged



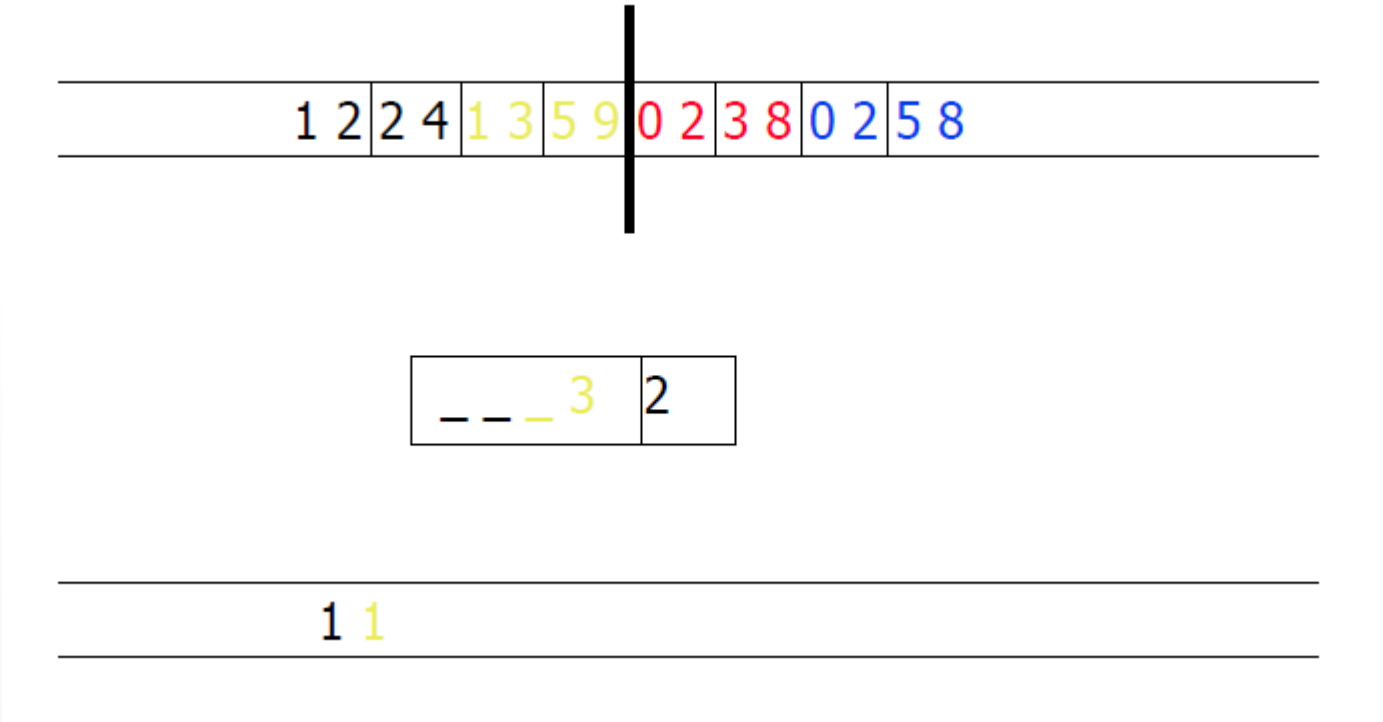
External Sort-Merge - primer



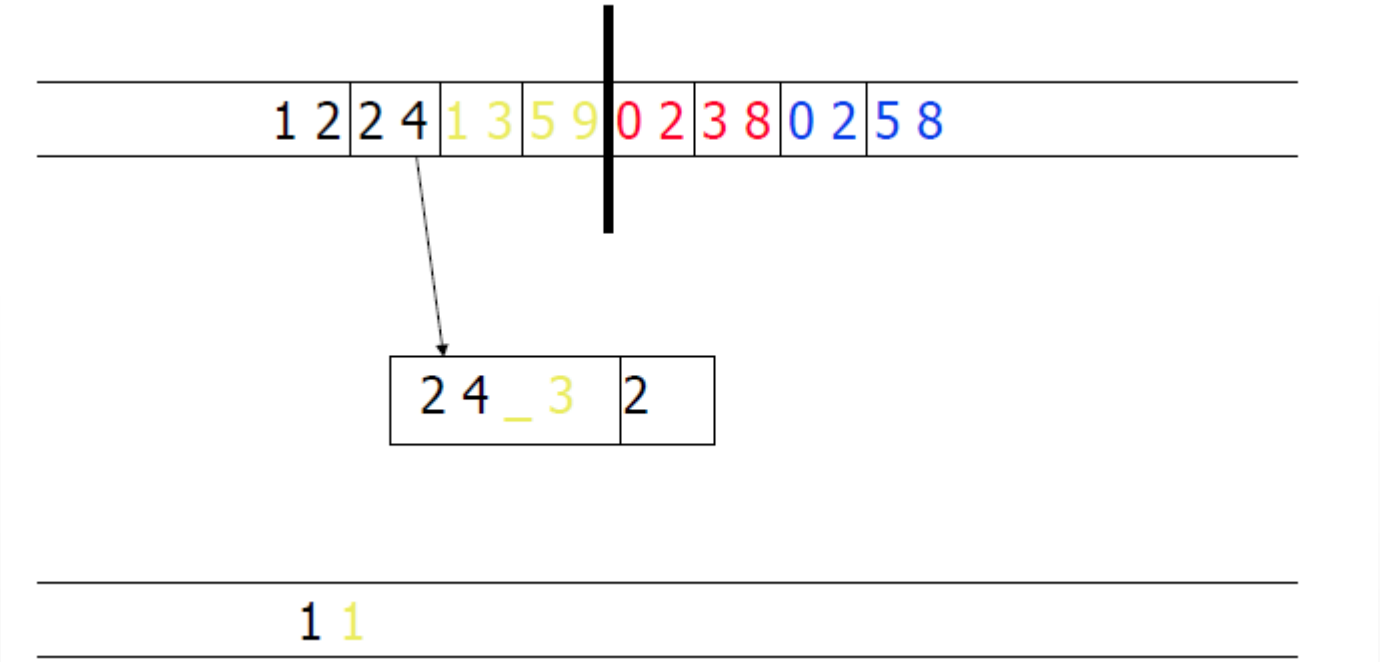
External Sort-Merge - primer



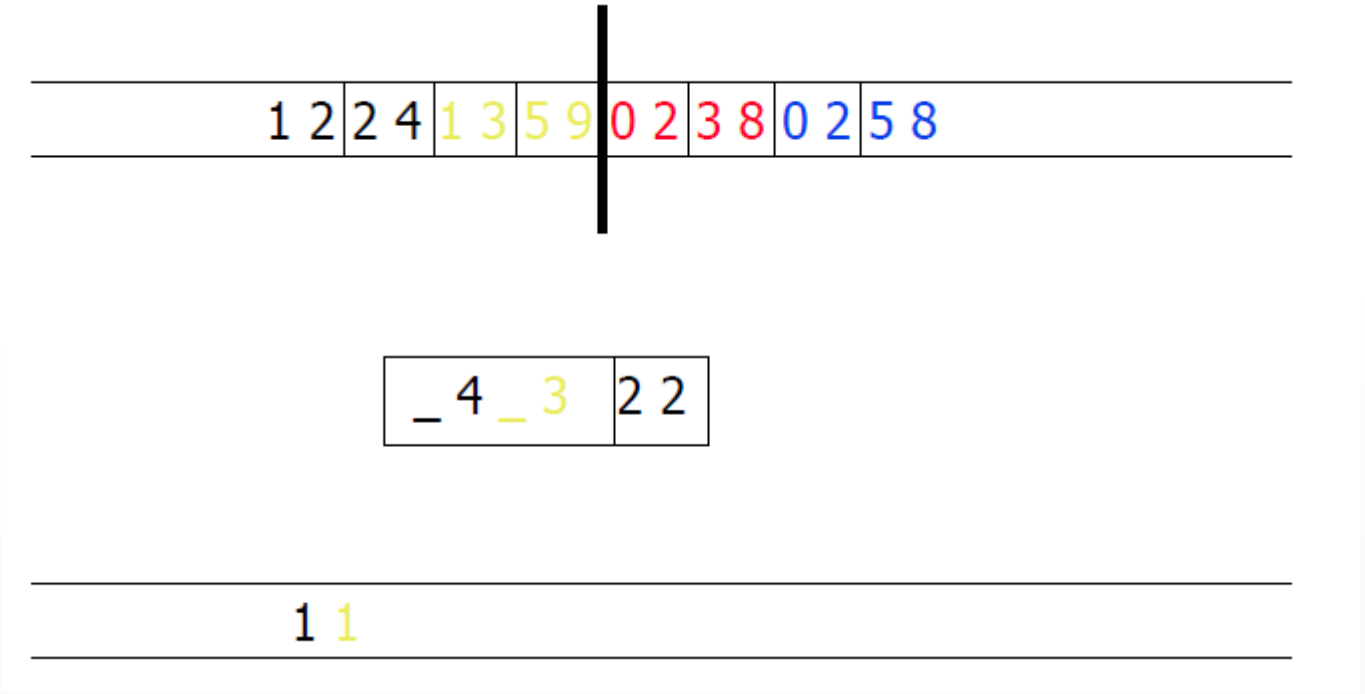
External Sort-Merge - primer



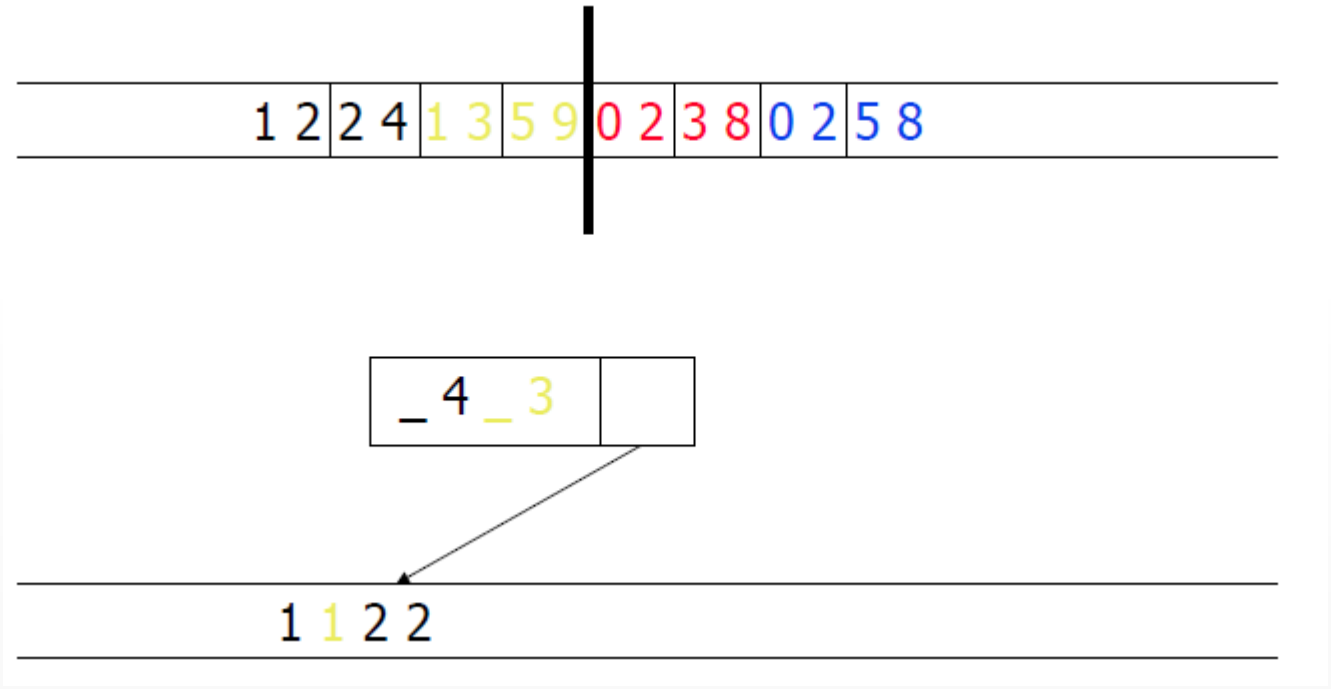
External Sort-Merge - primer



External Sort-Merge - primer

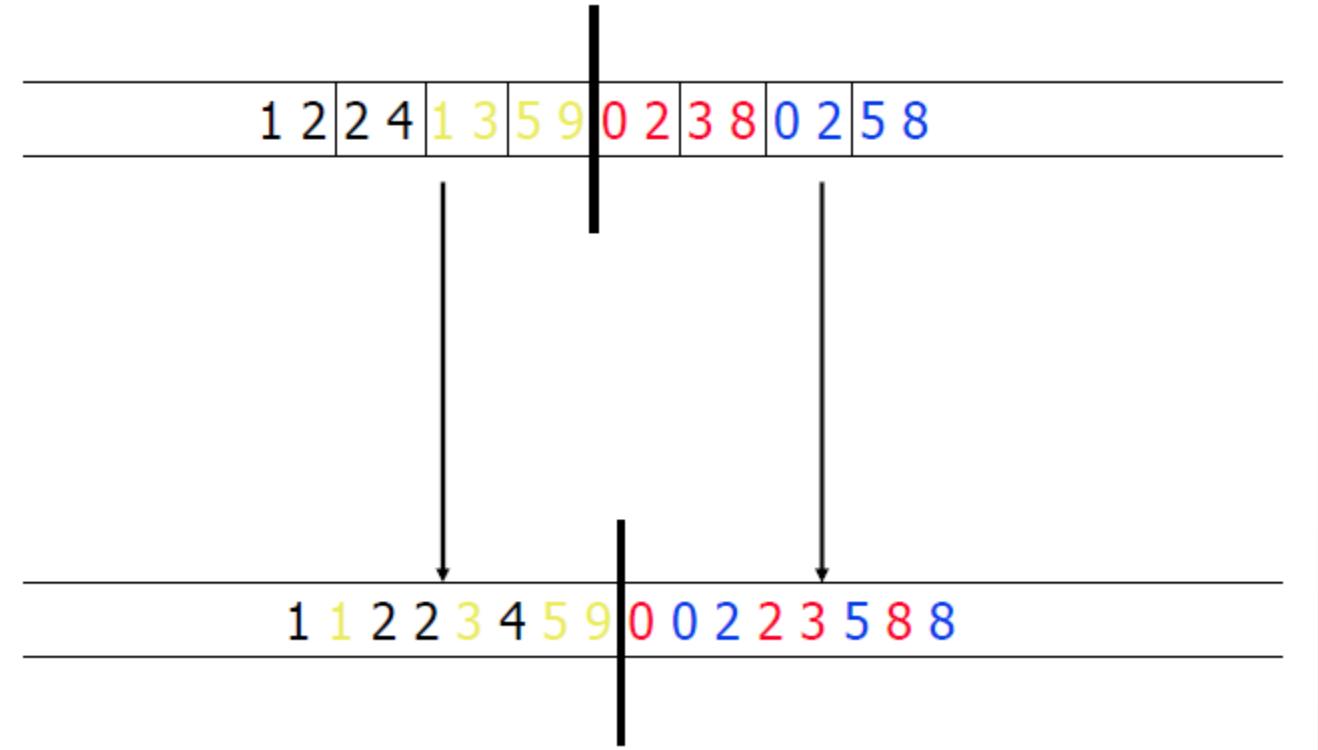


External Sort-Merge - primer



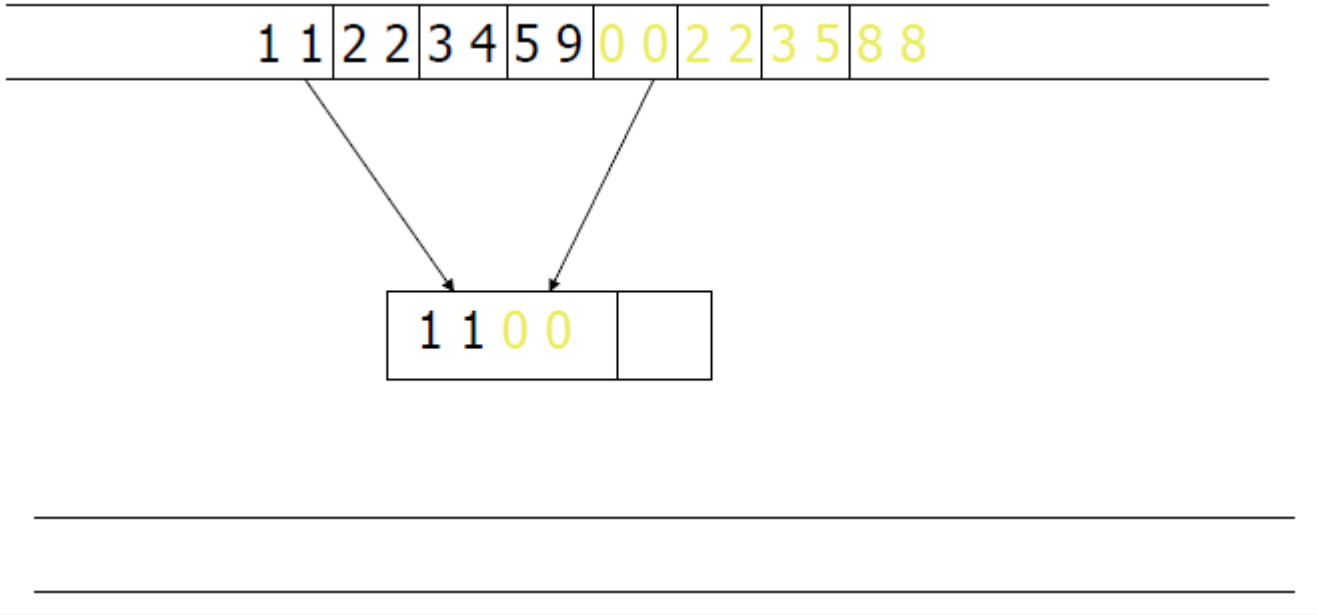
External Sort-Merge - primer

Merging Runs End of First Pass



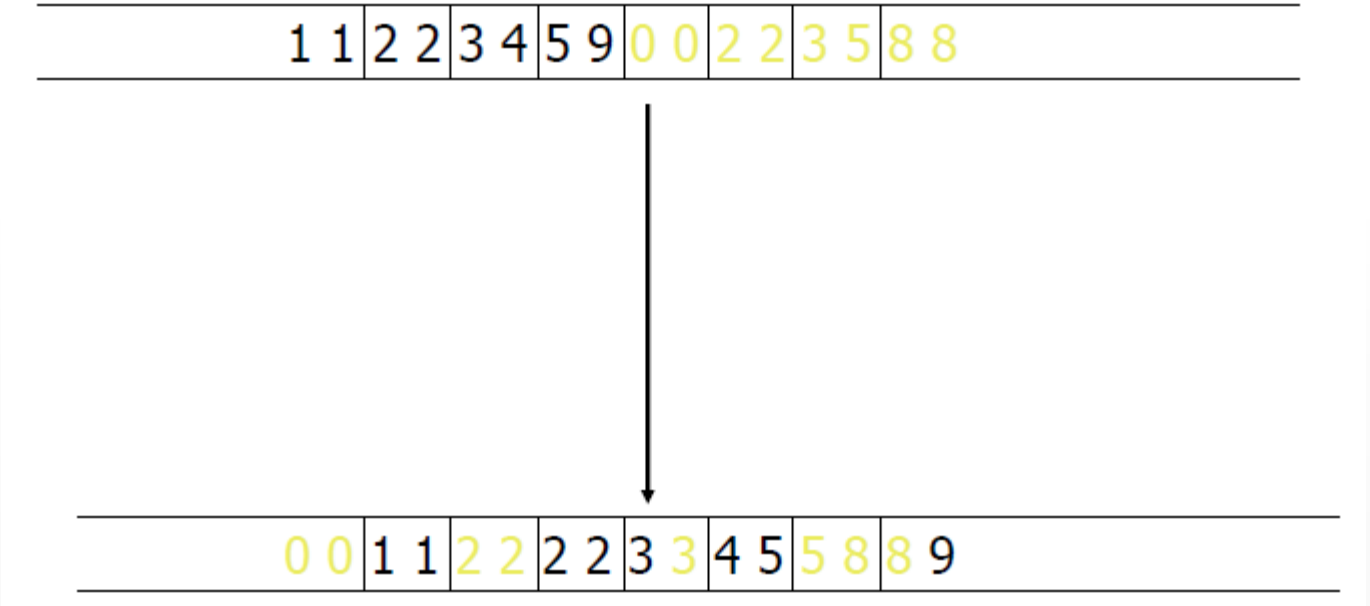
External Sort-Merge - primer

Second Phase: Merging Runs Second Pass



External Sort-Merge - primer

End of Second Pass



General External Sort-Merge Cost

- M = number of main memory page buffers
- b_r = number of pages in file to be sorted
- Partial sort phase: sort M pages at a time; create N/M sorted runs on mass store, **cost** = b_r (r/w for every block)
- Merge Phase: merge all runs into a single run using M-1 buffers for input and 1 output buffer
  - Merge step: divide runs into groups of size M-1 and merge each group into a run; cost = 2N
  - each step reduces number of runs by a factor of M-1

  Cost of merge phase:
  - (N/M)/(M-1)k runs after k merge steps
  - [log M-1(N/M)] merge steps needed to merge an initial set of N/M sorted runs
  - cost = [2N $\log_{M-1}(N/M)$] ≈ 2N($\log_{M-1}$ N -1)
- Total **cost ≈ $b_r(2\log_{M-1}(b_r/M) + 1)$**

Procena troška fizičkog plana

Cena fizičkih planova

Procena troškova

- Efikasnost upita zavisi od primenjenih algoritama
  - Da bi se razumeo plan izvršavanja upita treba poznavati algoritme
  - Razumenti plan izvršenja da bi se podesio DBMS
- U analizi efikasnosti celog plana izvršenja osnovni korak je procena troškova operatora.
- Definisanje *cost* modela – modela troškova
  - Poređenje algoritma
  - Upotreba u procesu optimizacije plana izvršavanja

Funkcija troška - komponente

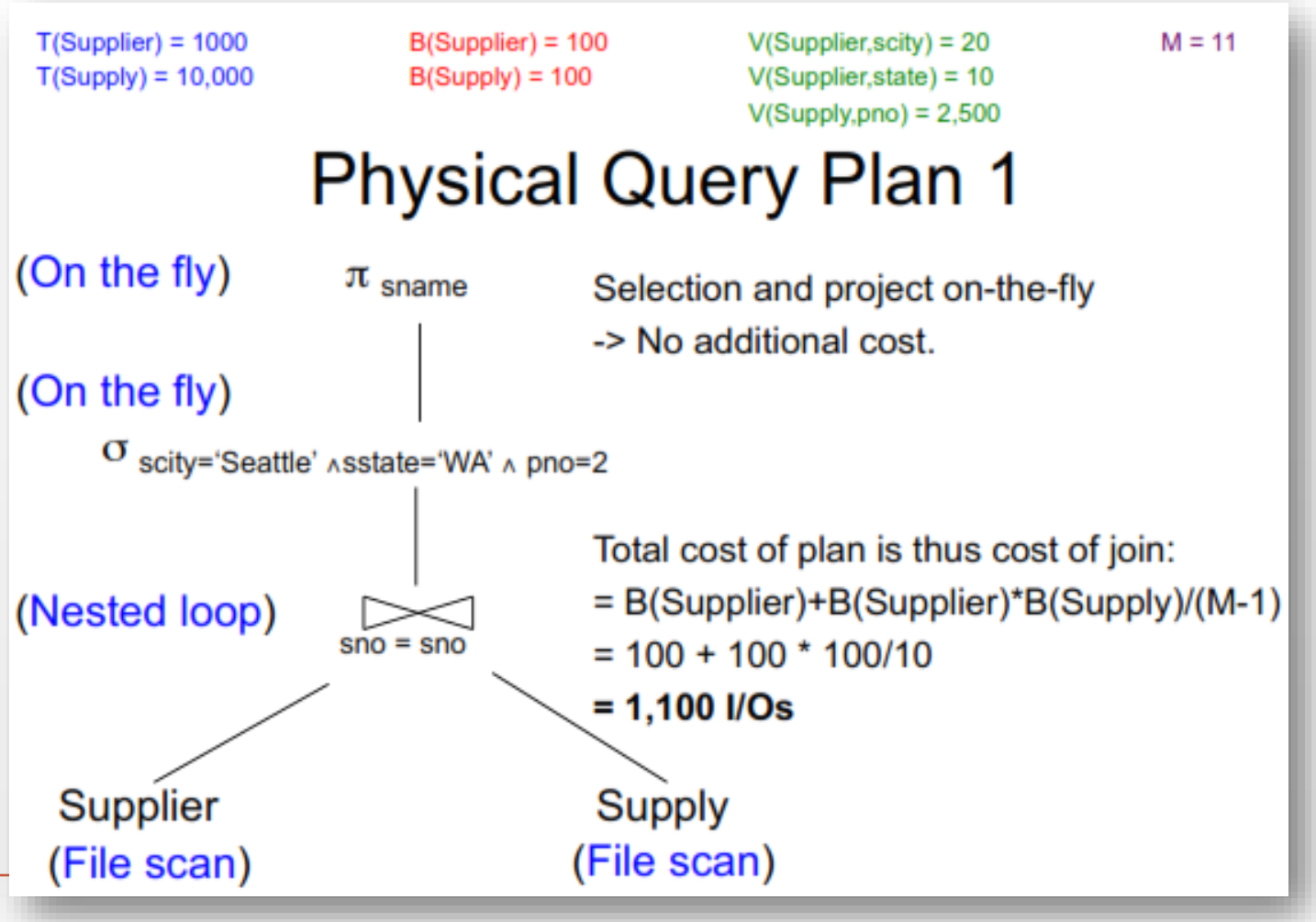
- Trošak pristupa spoljnoj memoriji – dominantan u bazama sa velikim kolekcijama podataka.
- Trošak sladištenja međurezultata
- Trošak računanja – CPU, presudna kod malih, in-memory baza i sistema
- Troškovi komunikacija

- Najčešće se optimizacija sprovodi na osnovu funkcije troška pristupa spoljnoj memoriji.

Troškovi pristupa spoljnoj memoriji

- Broj operacija pretrage
- Broj čitanja blokova
- Broj pisanja blokova
- Cena je veoma zavisna od veličine prostora u bafer pulu koji je dodeljen operatoru.

Troškovi fizičkog plana
primer



Enumeracija fizičkih planova

System R optimizator - Selinger style

Seliger style optimizatori – dominantna vrsta optimizatora za baze koje nisu distribuirane i ne spadaju u domen big data sistema.

Karakteristike optimizatora R sistema (glavni koncept postavila Patricia Selinger):

- Upotreba statističkih podataka o objektima baze u procesu ocene planova.
- Razmatranje samo onih planova u čijim je operacijama spajanja obezbeđeno da je unutrašnja relacija bazna.
- Usmeravanje optimizacije na upite bez ugnježdavanja i ad hoc tretiranje ugnježdenih upita
- Nesprovodjenje eliminacije duplikata za operaciju projekcije (osim u poslednjem koraku u kojem se zahteva DISTINCT)
- Model ocene koji uključuje i CPU i I/O troškove.

Pristupi enumeraciji fizičkih planova

Osnovna ideja

- ekshaustija, ispitivanje celog prostora pretrage, koji se dobija svim mogućim scenarijima implementacije logičkih planova
- Svakom planu se pridružuje cena i bira se onaj sa najnižom cenom.

Dva pristupa u pretrazi prostora mogućih planova:

- Top-down (odozgo na dole)
  Za svaku moguću implementaciju operacije u korenu (logičkog) stabla se razmatraju svi mogući načini ocene argumenata opracije, računaju cene svake kombinacije i bira najbolja.
- Bottom-up (odozdo na gore)
  Za svaki podizraz u logičkom stablu, ocenjuju se sve moguće implementacije. Mogućnosti i cene za podizraz E se izračunavaju na osnovu njegovih podizraza i njihovim kombinovanjem.

Oba načina ispituju ceo prostor, pa se ne razlikuju ukoliko se ne uključe heuristike i smanji prostor pretrage.

Heuristička selekcija

Princip koji se primenjuje i na logičke planove.

Neke od mogućih su:

- Spajanje para relacija čiji rezultat ima najmanju cenu, pa se kao takav uključuje u spajanje sa sledećom relacijom
- Ako se implementira selekija A=a nad atributom A koji je indeksiran, onda koristiti index-scan
- Ako argument spajanja ima indeks nad atributom spajanja onda koristiti index-join sa tom relacijom (tim argumentom) u unutrašnjoj petlji
- Ako je jedan argument spajanja sortiran po atributu spajanja, dati prednost sort-join u odnosu na hash-join algoritam
- Pri izračunavanju unije ili preseka tri ili više relacija, grupisati prvo najmanje.
- Svakom planu se pridružuje cena i bira se onaj sa najnižom cenom.

Branch-and-Bound enumeracija

Ovaj pristup počinje upotrebom heuristike za pronalaženje dobrog fizičkog plana za ceo logički plan (određivanje početnog plana), a zatim pokušava da ga popravi.

- Neka je cena tog plana C.
- Zatim se razmatraju planovi za delove/podupite.
- Odbacuju se svi planovi podupita koji imaju cenu veću od C.
- Ako se zamenom implementacije podupita konstruiše plan čija je ukupna manja od C, tada se taj plan proglašava tekućim koji dalje treba popravljati.
- Ova pretraga se može zaustaviti u bilo kom trenutku.

Hill Climbing

I ovaj postupak kreće od heuristički odabranog fizičkog plana.

- Na plan se primenjuju male izmene, npr. promena načina izvršavanja jednog operatora, promena redosleda izvršavajna join operacija.
- Vrši se ocena tako imenjenog plana.
- Pretraga se zaustavlja kad nikakve male izmene ne dovode do smanjenja cene.

Dinamičko programiranje

Bottom-up strategija u kojoj se za svaki podizraz zadržava plan sa najmanjom cenom.

Selinger-Style optimizacija

Slično prethodnom, pri čemu se ne pamti samo jedan plan za svaki podizraz, već i one koji rezultuju rezultatom sortiranim na način koji može biti koristan u realizaciji operacija u višim delovima stabla, npr. sortiran po atributu:

- Na koji se u korenu takođe primenjuje sortiranje
- Po kojem se kasnije vrši grupisanje
- Po kojem se kasnije vrši spajanje

Dalje

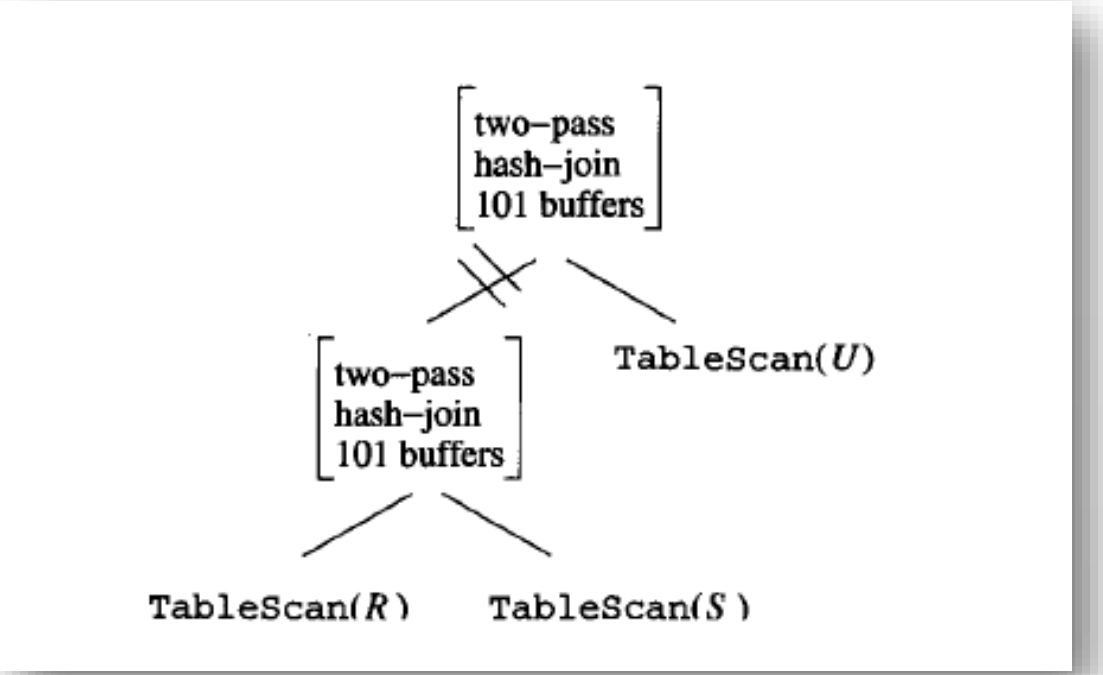
Nakon odabira osnovnog kostura fizičkog plana potrebno je i:

- Odrediti tretman međurezultata – materijalizacija ili pipeline
- Beleženje operatora fizičkog plana, koji uključuju i detalje koji se tiču metoda pristupa relacijama i konkretnih algoritama za primenu relacionih operatora.

Zapis fizičkog plana

Fizički plan sadrži

- Operacije koje se izvode
- Neophodni parametri, npr. uslov u teta spajanju.
- Detalji algortima i broj prolaza
- Procenjen broj bafera



Redosled izvršavanja fizičkih operacija

Fizički plan je stablo čije se operacije moraju poređati u neki redosled izvršavanja.

Postupak uređivanja:

- Podeli (preseci) stablo na podstabla na svakoj grani koja ima materijalizaciju. Podstabla će se izvršavati jedno po jedno.
- Redosled se određuje po principu odozdo na gore i sa leva na desno
- Izvrši sve čvorove svakog podstabla korišćenjem mreže iteratora. Svi čvorovi u podstablu se izvršavaju ‘simultano’ sa GetNext pozivima među operatorima kojima se definiše redosled.

https://github.com/rexshihaoren/SimpleDB