Optimizacija

Nenad Mitić
Matematički fakultet
nenad@matf.bg.ac.rs

Uvod

- Problem: kako izabrati efikasnu strategiju za izračunavanje postavljenog upita?
- Optimizacija upita predstavlja i izazov i mogućnost u relacionim bazama podataka
  - izazov jer je potrebna radi dostizanja prihvatljivih performansi sistema
  - mogućnost jer ilustruje prednost relacionog pristupa (u odnosu na nerelacione)

Uvod

- Optimizacija se vrši automatski, bez intervencije korisnika
- Program za optimizaciju će bolje uraditi optimizaciju od korisnika relacionog sistema
  - dobar optimizator ima na raspolaganju statističke informacije iz sistemskog kataloga
  - ako se statistika promeni moguće je da treba izvršiti reoptimizaciju - redak slučaj u slučaju ručne optimizacije od strane korisnika

Uvod

Program za optimizaciju će bolje uraditi optimizaciju od korisnika relacionog sistema (nastavak)

- optimizator je program i prema samoj definiciji je strpljiviji od uobičajenog ljudskog korisnika. Optimizator je, u odnosu na čoveka, sposoban da pregleda stotine različitih strategija pristupa za dati upit
- u optimizator su uključene veštine i znanje "najboljih"ljudskih programera. Posledica toga je da su te osobine svima na raspolaganju

Primer

Prikazati imena studenata koji su polagali ispit iz RBP (id=2016)

```
((dosije join ispit)
where Id_predmeta(2016)){ime}
```

- Neka baza sadrži 100 studenata i podatke o 10000 ispita od kojih se 50 odnosi na predmet RBP
- Neka se Dosije i Ispit nalaze direktno na disku, svaki relvar u po jednoj datoteci sa jednom torkom u jednom slogu
- Kriterijum efikasnosti - broj čitanja i pisanja po disku, odnosno broj U/I operacija

Primer - nastavak

Direktno izračunavanje

1. Izvršiti spajanje Dosije i Ispit (preko Indeks)
   - čita se 10000 ispita za svakog od 100 studenata
   - međurezultat se sastoji od 10000 spojenih torki koje se pišu natrag na disk (jer memorija nije dovoljno velika da ih primi)
2. Vrši se restrikcija rezultata u koraku 1) na torke koje sadrže 2016
   - čita se ponovo 10000 torki sa diska
   - rezultat je 50 torki koje mogu da ostanu u memoriji
3. Projektuju se rezultati iz koraka 2) preko IME
   - najviše 50 torki koje ostaju u memoriji

Primer - nastavak

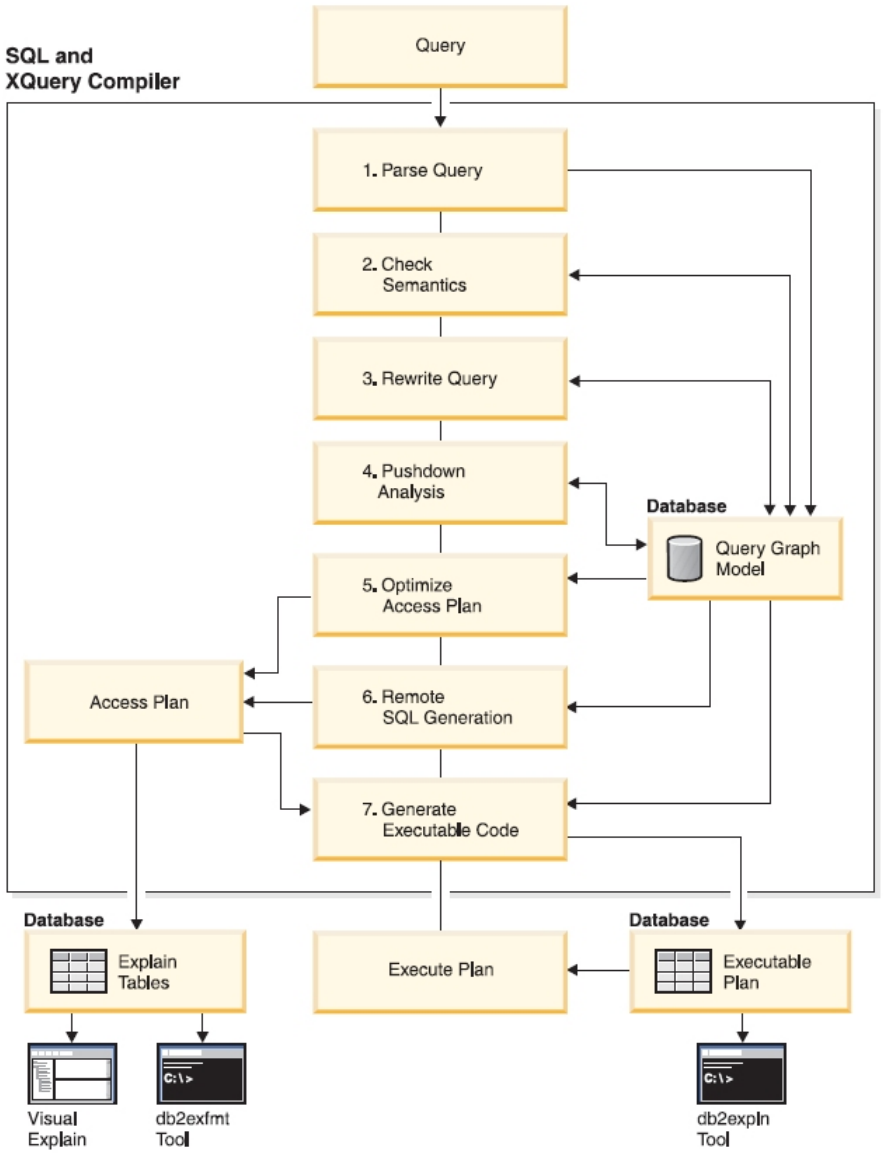
Izračunavanje sa optimizacijom

1. Izvršiti restrikciju na torke koje sadrže 2016
   - čita se 10000 ispita za svakog od 100 studenata
   - međurezultat se sastoji od 50 torki koje ostaju u memoriji
2. Spojiti rezultate koraka 1) (preko Indeks) sa Dosije
   - čita se 100 studenata
   - dobijeni rezultat ima ponovo 50 torki koje ostaju u memoriji
3. Projektuju se rezultati iz koraka 2) preko IME
   - najviše 50 torki koje ostaju u memoriji

Primer - nastavak

- Prvi pristup ima ukupno 1030000 U/I torki dok drugi ima samo 10100
- Razlika u dužini izvršavanja je očigledna - drugi pristup daje oko 100 puta brže izračunavanje
- Moguća su i dalja poboljšanja npr. ako se atributi indeksiraju (npr. Id_predmeta)

Faze obrade upita - shema



Faze obrade upita

U obradi upita se mogu identifikovati osnovne faze:

- Parsiranje upita i provera semantike
- Konverzija upita u kanonički oblik
- Analiza i izbor kandidata za procedure niskog nivoa
- Formiranje planova upita i izbor najjeftinijeg

Parsiranje upita i provera semantike

- Provera sintaksne ispravnosti upita
- Prevođenje upita na interni zapis
- Provera korektnosti tipova argumenata, funkcija, korelacija, podupita, ...

Prevođenje upita na interni zapis

- Početni upit se prevodi u internu reprezentaciju koja je pogodnija za obradu u računaru
- Obično se koristi drvo upita ili drvo apstraktne sintakse
- Za naše potrebe izabraćemo pogodniji formalizam za predstavljanje upita, npr. relacionu algebru
- ```
  ((dosije join ispit)
  where Id_predmeta(2016)){ime}
  ```
  ```

Prevođenje upita na interni zapis

Konverzija upita u kanonički oblik

- U ovoj fazi optimizator obavlja operacije za koje “postoji garancija da su dobre”, bez obzira kakvi su podaci i koja baza u pitanju
- Npr. SQL upit je moguće zapisati na više načina koje pre dalje obrade treba dovesti na ekvivalentan kanonički oblik koji je mnogo efikasniji
- Za konverziju upita u kanonički oblik optimizator koristi različita pravila za transformaciju
- Dva upita q_1 i q_2 su ekvivalentni ako i samo ako se, pri njihovom izvršavanju, u svim slučajevima dobija isti rezultat.

Konverzija upita u kanonički oblik

Kanonički oblik (definicija):

- Za podskup C datog skupa upita Q se kaže da je u kanoničkoj formi za Q ako i samo ako je svaki upit q iz Q ekvivalentan nekom upitu c iz C
- Upit c predstavlja kanonički oblik upita q

Posledica: sve “korisne” osobine koje mogu da se primene na upit q važe i za upit c. Zbog toga je dovoljno da se razmatra manji skup upita C umesto većeg Q da bi se dobili različiti “korisni” rezultati.

Analiza i izbor kandidata za procedure niskog nivoa

- Posle konverzije i predstavljanja u kanoničkom obliku optimizator mora da odluči na koji način će izvršavati transformisani upit
- Osnovna strategija je posmatranje izraza koji predstavlja upit kao niza operacija niskog nivoa (spajanje, projekcija, restrikcija, ...) između kojih postoje određene zavisnosti npr. projekcija obično zahteva da ulazne torke budu sortirane što znači da rezultat prethodne operacije treba da bude sortiran
- Optimizator razmatra postojanje indeksa, postojanje pristupnih puteva, fizičku distribuciju podataka, ...

Analiza i izbor kandidata za procedure niskog nivoa

- Za svaku od operacija niskog nivoa optimizator ima na raspolaganju skup predefinisanih procedura za njihovu implementaciju
- Svaka procedura ima pridruženu (parametrizovanu) formulu za određivanje cene koštanja, obično u zavisnosti od U/I operacija na disku, CPU vremena, veličine međurezultata,...
- Na osnovu informacija iz kataloga o tekućem stanju baze i međusobnih zavisnosti operacija niskog nivoa optimizator bira jednu ili više procedura za implementaciju svake od operacija niskog nivoa

Formiranje planova upita i izbor najjeftinijeg

- Formira se skup kandidata na plan upita između kojih se bira najbolji (tj. najjeftiniji)
- Svaki plan je kombinacija kandidata za implementaciju procedura za operacije niskog novoa
- Cena upita je jednaka zbiru cena pojedinačnih procedura
- Problem je određivanje cene upita jer formule zavise od veličine relacija koje se obrađuju

Formiranje planova upita i izbor najjeftinijeg

- Svi sem najjednostavnijh upita uključuju formiranje međurezultata pri izvršavanju tako da najveći deo parametara nije unapred poznat
- Zbog toga optimizator pravi procene cene koštanja upita
- Primer procene: Visual Explain u DB2
- Alati za procenu performansi i analizu mera za poboljšanje
- Primer: Optim Query Workload Tuner

Transformacija izraza

Restrikcija i projekcija

1. (A WHERE restrikcija1) WHERE restrikcija2
   $\Longleftrightarrow$ A WHERE restrikcija1 AND restrikcija2
2. (A {atributi1}) {atributi2} $\Longleftrightarrow$ A {atributi2}
3. (A {atributi}) WHERE restrikcija $\Longleftrightarrow$ (A WHERE restrikcija) {atributi}

Primedba: u opštem slučaju je dobro primeniti restrikciju pre projekcije jer se time smanjuje veličina ulaza u projekciju čime se smanjuje veličina podataka koje treba sortirati radi eliminisanja duplikata

Transformacija izraza

Distribucija

1. Za unarni operator f se kaže da je distributivan preko binarnog operatora O ako i samo ako važi $f(AOB) \equiv f(A)Of(B)$
2. Restrikcija je distributivna
   - preko unije, preseka i razlike
   - preko spajanja ako i samo ako se uslov restrikcije sastoji od najviše dva odvojena uslova restrikcije koja su spojena konjunkcijom (AND), jedan za svaki operand u spajanju

Transformacija izraza

Projekcija je distributivna

1. preko unije i preseka
   $(A\ UNION\ B)\{C\} \equiv A\{C\}\ UNION\ B\{C\}$
   $(A\ INTERSECT\ B)\ \{C\} \equiv A\{C\}\ INTERSECT\ B\{C\}$
2. ne i preko razlike
3. preko spajanja
   $(A\ JOIN\ B)\{C\} \equiv (A\ \{AC\})\ JOIN\ (B\ \{BC\})$
4. ako i samo ako
   - AC je unija (a) atributa koji su zajednički sa A i B i (b) onih atributa C koji se pojavljuju samo u A
   - BC je unija (a) atributa koji su zajednički sa A i B i (b) onih atributa C koji se pojavljuju samo u B

Transformacija izraza

Komutativnost

1. Binarni operator O je komutativan ako i samo ako važi $AOB \equiv BOA$
2. Unija, presek i spajanje su komutativni
3. Razlika i delenje nisu
4. Posledica: ako upit uključuje spajanje dve relacije A i B komutacija omogućuje da se "manja"relacija uzme za spoljašnju

Transformacija izraza

Idempotencija

1. Binarni operator O je idempotentan ako i samo ako važi $AOA \equiv A$
2. Unija, presek i spajanje su idempotentni
3. Razlika i delenje nisu
4. Osobina idempotencije može da bude korisna u transformaciji izraza

Skalarni izračunljivi izrazi

Optimizator mora da vodi računa i o transformacijama aritmetičkih izraza (npr. komutacija, asocijacija, distribucija) jer na ovaj tip izraza može da se naiđe u kontekstu operatora extend i summarize.

Transformacija izraza

Logički izrazi. Na primer,

1. izraz $A > B$ *and* $B > 3$ se može, na osnovu tranzitivnosti operatora $>$, transformisati u izraz
   $A > B$ *and* $B > 3$ *and* $A > 3$
2. Transformacija je korisna jer omogućuje dodatnu restrikciju na A pre izvođenja spajanja (sa $>$). Ideja izvođenja ranije restrikcije je pogodna i primenjuje je više komercijalnih produkata
3. $A > B$ *or* $(C = D$ *and* $E < F)$ se može transformisati u $(A > B$ *orC* $= D)$ *and* $(A > B$ *or* $(E < F)$

Transformacija izraza

Svaki logički izraz se može transformisati u ekvivalentni izraz u konjuktivnoj normalnoj formi (KNF)

1. KNF je izraz oblika C_1 AND C_2 AND ... AND C_n, pri čemu ni jedan od izraza C_i ne sadrži konjunkciju (AND)
2. Prednost KNF je što je izraz tačan ako su svi konjunkti tačni, a netačan ako je bar jedan od njih netačan
3. Kako je konjunkcija komutativna optimizator može da bira redosled izvršavanja konjukata idući od jednostavnijiih ka složenijima
4. KNF je pogodna kod sistema sa paralelnom obradom

Transformacija izraza

Semantičke transformacije

1. U izrazu (DOSIJE JOIN ISPIT) {OCENA} spajanje se vrši uparivanjem spoljašnjeg ključa (sa jedne strane) i kandidata za ključ (sa druge strane). Odatle sledi da se svaka torka iz tabele ISPIT spaja sa nekom torkom iz tabele DOSIJE i da zbog toga svaka torka iz tabele ISPIT daje neku vrednost za atribut OCENA u krajnjem rezultatu. Odavde se vidi da nema potrebe za spajanjem i da izraz može biti uprošćen u ISPIT {OCENA}
2. Ovakav tip transformacije, iako je značajan, retko se sreće kod komercijalnih sistema zbog složenosti

Statistika u bazi podataka

Faze procesa optimizacije koriste statistiku baze podataka koja se čuva u katalogu

1. statistika o osnovnim tabelama
2. statistika o svakoj koloni u osnovnoj tabeli
3. statistika o indeksima
4. statistika se ne sakuplja automatski već na zahtev korisnika
5. RUNSTATS naredba u DB2

Implementacija operatora spajanja

U najvećem broju slučajeva sistem ima potrebu da vrši grupisanje torki prema zajedničkim vrednostima u određenim atributima. Za grupisanje se koriste različite tehnike. Na primer, za spajanje:

1. Gruba sila (eng. brute force) u kojoj se prave sve moguće kombinacije torki u spajanju
2. Pomoću indeksa koji se koristi za direktan pristup uparenim torkama unutrašnje relacije spajanja
3. Pomoću heša koji se koristi umesto indeksa za direktan pristup uparenim torkama unutrašnje relacije spajanja

Implementacija operatora spajanja

U najvećem broju slučajeva sistem ima potrebu da vrši grupisanje torki prema zajedničkim vrednostima u određenim atributima. Za grupisanje se koriste različite tehnike. Na primer, za spajanje:

1. Mešanjem relacija koje su fizički sačuvane u redosledu atributa po kome se spajaju
2. Heš tehnika koja omogućuje jedan prolaz kroz obe relacije koje se spajaju
3. kombinacijom ovih tehnika

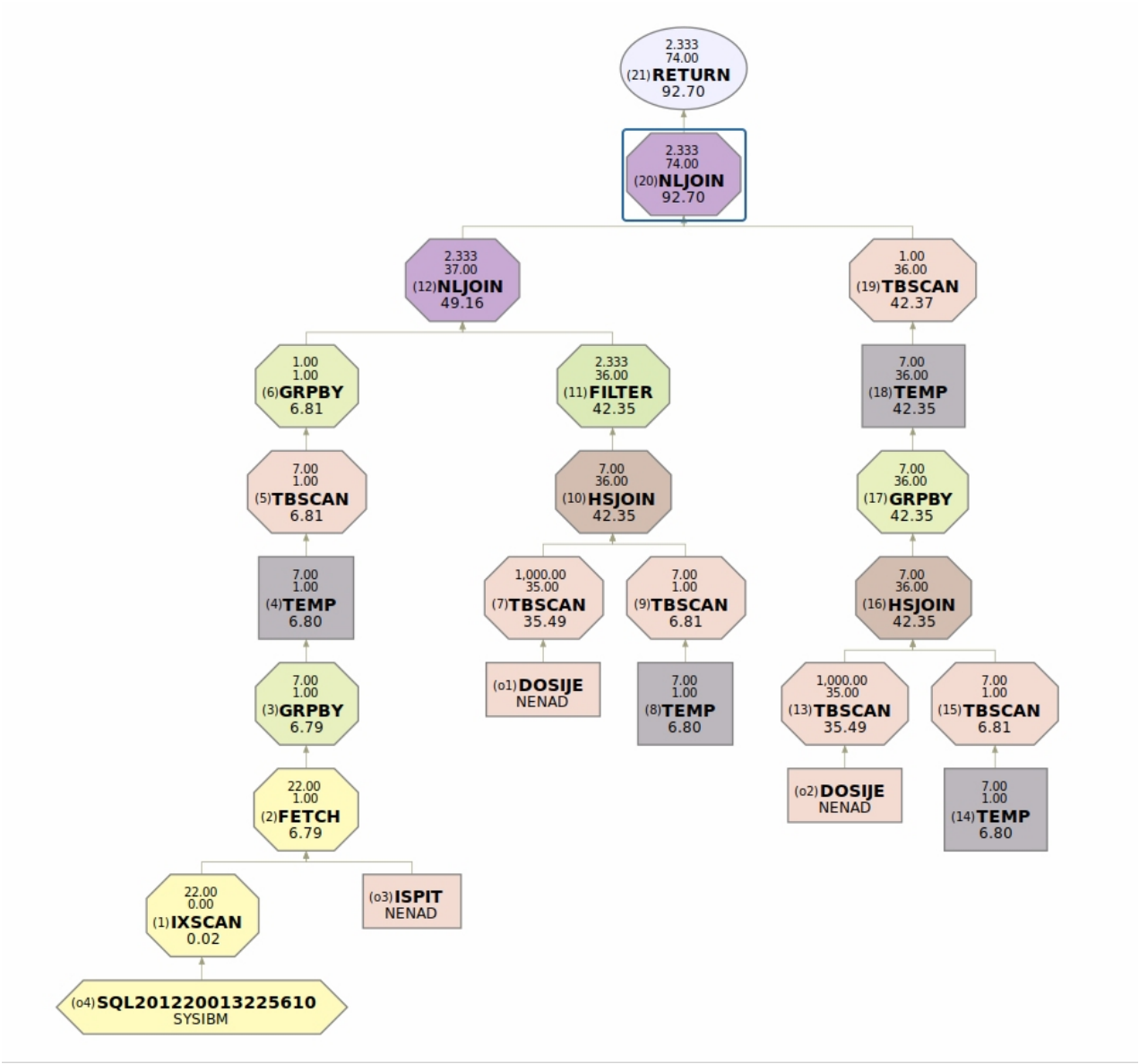
Detaljne informacije o procedurama niskog nivoa se nalaze u *DB2 Performance Tuning* i na adresi
https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.admin.explain.doc/doc/r0052023.html

Optimizacija u Db2

SQL upit u DB2 je skup šelect-from-where"blokova

1. Bira se redoled blokova, pri čemu se u slučaju ugneždenih blokova optimizuje prvo koji je na najvećoj dubini (naj-unutrašnjiji blok)
2. EXPLAIN – alat za procenu cene upita
3. Korisnik DB2 LUW na raspolagaju ima Visual Explain – alat za procenu sa grafičkim interfejsom
4. Potrebno je izvršiti inicijalizaciju EXPLAIN tabela preko Data Studija ili sa komandne linije (direktorijum MISC, datoteka EXPLAIN.DDL

Optimizacija u Db2



Nivoi optimizacije u Db2

Nivoi optimizacije u Db2

1. 0 - Use a minimal amount of optimization
2. 1 - Use a degree of optimization roughly comparable to DB2/6000 Version 1, plus some additional low-cost features not found in Version 1
3. 2 - Use features of opt level 5, but simplified join algorithm
4. 3 - Perform a moderate amount of optimization; similar to the query optimization characteristics of DB2 for z/OS
5. 5 - Use a significant amount of optimization; with Heuristic Rules (default)
6. 7 - Use a significant amount of optimization; without Heuristic Rules
7. 9 - Use all available optimization techniques

Uvod

- Isti zadatak može da se reši na više načina
- Rešenja mogu da se razlikuju po efikasnosti - kako izabrati najefikasnije
- 16.primeri.sql - različita rešenja istih zadataka (dva primera)
- Proceniti efikasnost sa Visual explain - koji su razlozi razlika?
- *SARGable - Search ARGument* atributi - atributi po kojima može da se vrši pretraživanje
- Neke od navedenih pravila optimizator može da transformiše - videti Visual Explain
- Pravila za prevođenje predikata su prikazana DB2 V11.5 Performance Tuning, Tabela 61

Efikasnost SELECT naredbe

- Navesti samo atribute koji su neophodni - ne koristiti "*"ako nema potrebe
- Koristiti predikate koji prave restrikciju samo na one slučajeve koji su potrebni
- Ako je potreban značajno manji broj slogova broja postojećih u tabeli koristiti OPTIMIZE FOR klauzulu
- Koristiti FOR READ ONLY/FOR FETCH ONLY klauzule
- Isključiti DISTINCT/ORDER BY gde nisu neophodni
- Koristiti UNION ALL umesto UNION gde je to moguće

Efikasnost SELECT naredbe

- Izbegavati konverziju numeričkih tipova
- Atributi koji se porede treba da budu istog tipa
- Ako je moguće, koristiti sledeće tipove podataka
  - CHAR umesto VARCHAR za kraće atribute
  - Integer umesto FLOAT, DECIMAL ili DECFLOAT
  - DECFLOAT umesto DECIMAL
  - Datumsko-vremenski tip umesto karaktera
  - Brojačane vrednosti umesto karaktera

Efikasnost SELECT naredbe

- Sem u slučaju malih tabela izbegavati `SELECT count(*) from <tabela>` za proveru da li je tabela prazna
- Koristiti IN listu ako se isti atribut javlja u više predikata
- Ako je moguće, izbeći korišćenje OR predikata pri spajanju tabela
- ...

Neka pravila za kodiranje

Izbegavati, ukoliko je moguće, korišćenje skalarnih funkcija nad atributima u predikatu

Umesto

```
select ime,prezime
from   dosije
where  year(datum_rodjenja)=2002
```

efikasniji zapis je

```
select ime,prezime
from   dosije
where  datum_rodjenja between '2002-01-01' and '2002-12-31'
```

Neka pravila za kodiranje

Isključiti, ukoliko je moguće, primenu matematički funkcija nad atributima u predikatu

Umesto

```
select indeks,id_predmeta,ocena
from   ispit
where  godina_roka+5 > 2010
```

efikasniji zapis je

```
select indeks,id_predmeta,ocena
from   ispit
where  godina_roka > 2010 - 5
```

Neka pravila za kodiranje

Isključiti, DISTINCT kada god je to moguće. Ako treba eliminisati duplikate

- koristiti GROUP BY koji može da koristi indekse (ako postoje) radi eliminisanja sortiranja
- napisati upit upotrebom IN ili EXISTS. Korisno ako tabela koja vraća duplikate ne vraća podatke za neke vrednosti

Neka pravila za kodiranje

Umesto

```
select distinct id_predmeta, a.godina_roka
from   ispit a, ispitni_rok b
where  a.oznaka_roka=b.oznaka_roka
```

efikasniji zapis je

```
select id_predmeta, a.godina_roka
from   ispit a, ispitni_rok b
where  a.oznaka_roka=b.oznaka_roka
group  by id_predmeta, a.godina_roka
```

Neka pravila za kodiranje

Umesto

```
select id_predmeta, a.godina_roka
from   ispit a
where  a.oznaka_roka in (select oznaka_roka
                         from   ispitini_rok)
```

efikasniji zapis je

```
select id_predmeta, a.godina_roka
from   ispit a
where  exists (select 1
               from   ispitni_rok b
               where  b.oznaka_roka=a.oznaka_roka
              )
```

Neka pravila za kodiranje

Ne tražiti podatke koji su već poznati

Umesto

```
select indeks,id_predmeta,godina_roka,ocena
from   ispit
where  godina_roka=2020
```

efikasniji zapis je

```
select indeks,id_predmeta,ocena
from   ispit
where  godina_roka=2020
```

Neka pravila za kodiranje

Koristiti CASE umesto UNION, ako je moguće

Umesto

```
select creator,name,'Tabela'
from   sysibm.systables
where  type='T'
UNION
select creator,name,'Pogled'
from   sysibm.systables
where  type='V'
UNION
select creator,name,'Alias'
from   sysibm.systables
where  type='A'
UNION
select creator,name,'MQT'
from   sysibm.systables
where  type='S'
order by creator,name
```

Neka pravila za kodiranje

Efikasniji zapis je

```
select creator,name,
       case type
            when 'T' then 'Tabela'
            when 'V' then 'Pogled'
            when 'A' then 'Alias'
            when 'S' then 'MQT'
       end
from   sysibm.systables
order by creator,name
```

Neka pravila za kodiranje

CASE može da se koristi i u drugim naredbama - npr. Update

```
update ispit
set    ocena= case
              when bodovi>90 then 10
              when bodovi>80 then 9
              when bodovi>70 then 8
              when bodovi>60 then 7
              when bodovi>50 then 6
              else                5
       end
where godina_roka=2015
```

Neka pravila za kodiranje

- U kodiranju dati prednost *SARGable* atributima
- U kodiranju obratiti pažnju na konstrukciju predikata nad atributima gde je definisan indeks
- Tip atributa može se videi u Visual Explain pri odabiru procedure niskog nivoa
- Detaljniji prikaz u DB2 V11.5 Performance Tuning, Tabele 58 i 60
- *A Guide to Db2 Performance for Application Developers Code for Performance from the Beginning* - Craig S. Mullins
- Obratiti pažnju na procedure niskog nivoa - da li mogu da se promene

Procedure niskog nivoa

Naziv	Ulazna grana		Izlaz	Funkcija
	prva	druga		
IXSCAN	Indeksni čvor	—	Skup važećih identifikatora slogova (RID-ova)	Pretražuje indeks radi dobijanja važećih identifikatora slogova unutar zadatog intervala ključeva
FETCH	skup RID-ova	Čvor sa tabelom	Skup važećih slogova	Čita slogove podataka i odgovarajuće stranice na osnovu RID-a i primenjuje predikate ako postoje
TBSCAN	Tabela	—	Skup važećih slogova	Sekvencijano pretražuje ciljni prostor za čuvanje tabela radi dohvatanja stranica sa podacima i primenjuje predikate ako postoje
SORT	Skup slogova ili RID-ova	—	Skup sortiranih slogova ili RID-ova	Sortira ulazne podatke u okviru stranice (po RID-ovima) ili slogove (po ključevima)
NLJOIN (nested loop)	Skup slogova	Skup slogova	Skup slogova	Za svaki kvalifikovani slog iz prve (spoljašnje) tabele pretražuje drugu tabelu (unutrašnja) radi nalaženja uparenih slogova koji su rezultat spajanja
MSJOIN (merge scan)	Skup sortiranih slogova	Skup sortiranih slogova	Skup slogova	Pretražuje obe ulazne tabele radi nalaženja uparenih slogova koji su rezultat spajanja
HSJOIN (hash join)	Tabela	Tabela	Skup slogova	Formira se kod jednakosnog spajanja. Pretražuje se unutrašnja tabela, formira tabela za uparivanje na osnovu vrednosti atributa po kome se vrši spajanje, a zatim čita spoljašnja tabela, hešira atribut po kome se vrši spajanje i pretažuje tabela za uparivanje radi dobijanja rezultata