

Конкурентност

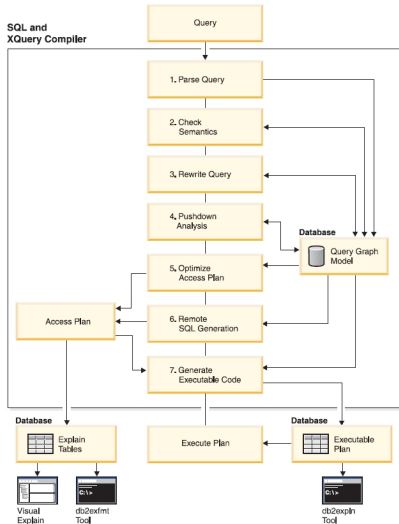
Ненад Митић

Математички факултет
nenad.mitic@matf.bg.ac.rs

Пример - наставак

- Први приступ има укупно 1030000 У/И торки док други има само 10100
- Разлика у дужини извршавања је очигледна - други приступ даје око 100 пута брже израчунавање
- Могућа су и даља побољшања нпр. ако се атрибути индексирају (нпр. `ld_predmeta`)

Фазе обраде упита - шема



Фазе обраде упита

У обради упита се могу идентификовати основне фазе:

- Парсирање упита и провера семантике
- Конверзија упита у канонички облик
- Анализа и избор кандидата за процедуре ниског нивоа
- Формирање планова упита и избор најјефтинијег

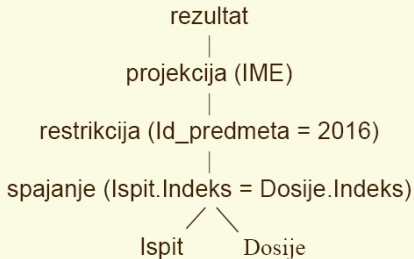
Парсирање упита и провера семантике

- Провера синтаксне исправности упита
- Превођење упита на интерни запис
- Провера коректности типова аргумената, функција, корелација, подупита, ...

Превођење упита на интерни запис

- Почетни упит се преводи у интерну репрезентацију која је погоднија за обраду у рачунару
- Обично се користи дрво упита или дрво апстрактне синтаксе
- За наше потребе изабраћемо погоднији формализам за представљање упита, нпр. релациону алгебру
- ```
((dosije join ispit)
where Id_predmeta(2016)){ime}
```

# Превођење упита на интерни запис



Algebra:  $((\text{Dosije JOIN Ispit}) \text{ WHERE } \text{Id\_predmeta}(2016))\{\text{IME}\}$

# Конверзија упита у канонички облик

- У овој фази оптимизатор обавља операције за које “постоји гаранција да су добре”, без обзира какви су подаци и која база у питању
- Нпр. SQL упит је могуће записати на више начина које пре даље обраде треба довести на еквивалентан канонички облик који је много ефикаснији
- За конверзију упита у канонички облик оптимизатор користи различита правила за трансформацију
- Два упита  $q_1$  и  $q_2$  су еквивалентни ако и само ако се, при њиховом извршавању, у свим случајевима добија исти резултат.

# Конверзија упита у канонички облик

Канонички облик (дефиниција):

- За подскуп  $C$  датог скупа упита  $Q$  се каже да је у каноничкој форми за  $Q$  ако и само ако је сваки упит  $q$  из  $Q$  еквивалентан неком упиту  $c$  из  $C$
- Упит  $c$  представља канонички облик упита  $q$

Последица: све “корисне” особине које могу да се примене на упит  $q$  важе и за упит  $c$ . Због тога је довољно да се разматра мањи скуп упита  $C$  уместо већег  $Q$  да би се добили различити “корисни” резултати.





# Анализа и избор кандидата за процедуре НИСКОГ НИВОА

- За сваку од операција ниског нивоа оптимизатор има на располагању скуп предефинисаних процедура за њихову имплементацију
- Свака процедура има придружену (параметризовану) формулу за одређивање цене коштања, обично у зависности од У/И операција на диску, CPU времена, величине међурезултата,...
- На основу информација из каталога о текућем стању базе и међусобних зависности операција ниског нивоа оптимизатор бира једну или више процедура за имплементацију сваке од операција ниског нивоа

# Формирање планова упита и избор најјефтинијег

- Формира се скуп кандидата на план упита између којих се бира најбољи (тј. најјефтинији)
- Сваки план је комбинација кандидата за имплементацију процедура за операције ниског новог
- Цена упита је једнака збиру цена појединачних процедура
- Проблем је одређивање цене упита јер формуле зависе од величине релација које се обрађују

# Формирање планова упита и избор најјефтинијег

- Сви сем најједноставнијх упита укључују формирање међурезултата при извршавању тако да највећи део параметара није унапред познат
- Због тога оптимизатор прави процене цене коштања упита
- Пример процене: Visual Explain и DB2
- Алати за процену перформанси и анализу мера за побољшање
- Пример: Query Tuner у Data Studio

# Трансформација израза

## Рестрикција и пројекција

- 1  $(A \text{ WHERE } \text{restrikcija1}) \text{ WHERE } \text{restrikcija2} \iff A \text{ WHERE } \text{restrikcija1} \text{ AND } \text{restrikcija2}$
- 2  $(A \{ \text{atributi1} \}) \{ \text{atributi2} \} \iff A \{ \text{atributi2} \}$
- 3  $(A \{ \text{atributi} \}) \text{ WHERE } \text{restrikcija} \iff (A \text{ WHERE } \text{restrikcija}) \{ \text{atributi} \}$

Примедба: у општем случају је добро применити рестрикцију пре пројекције јер се тиме смањује величина улаза у пројекцију чиме се смањује величина података које треба сортирати ради елиминисања дупликата





# Трансформација израза

## Комутативност

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

# Трансформација израза

## Идемпотенција

- 1 Бинарни оператор  $O$  је идемпотентан ако и само ако важи  $AOA \equiv A$
- 2 Унија, пресек и спајање су идемпотентни
- 3 Разлика и делење нису
- 4 Особина идемпотенције може да буде корисна у трансформацији израза



# Скаларни израчунљиви изрази

Оптимизатор мора да води рачуна и о трансформацијама аритметичких израза (нпр. комутација, асоцијација, дистрибуција) јер на овај тип израза може да се наиђе у контексту оператора `extend` и `summarize`.

# Трансформација израза

Логички изрази. На пример,

- 1 израз  $A > B$  and  $B > 3$  се може, на основу транзитивности оператора  $>$ , трансформисати у израз  $A > B$  and  $B > 3$  and  $A > 3$   
item Трансформација је корисна јер омогућује додатну рестрикцију на  $A$  пре извођења спајања (са  $>$ ). Идеја извођења раније рестрикције је погодна и примењује је више комерцијалних продуката
- 2  $A > B$  or  $(C = D$  and  $E < F)$  се може трансформисати у  $(A > B$  or  $C = D)$  and  $(A > B$  or  $(E < F)$

# Трансформација израза

Сваки логички израз се може трансформисати у еквивалентни израз у конјуктивној нормалној форми (KNF)

- 1 KNF је израз облика  $C_1 \text{ AND } C_2 \text{ AND } \dots \text{ AND } C_n$ , при чему ни један од израза  $C_i$  не садржи конјункцију (AND)
- 2 Предност KNF је што је израз тачан ако су сви конјункти тачни, а нетачан ако је бар један од њих нетачан
- 3 Како је конјункција комутативна оптимизатор може да бира редослед извршавања конјуктата идући од једноставнијих ка сложенијима
- 4 КНФ је погодна код система са паралелном обрадом



# Статистика у бази података

Фазе процеса оптимизације користе статистику базе података која се чува у каталогу

- 1 статистика о основним табелама
- 2 статистика о свакој колони у основној табели
- 3 статистика о индексима
- 4 статистика се не сакупља аутоматски већ на захтев корисника
- 5 RUNSTATS наредба у DB2

# Имплементација оператора спајања

У највећем броју случајева систем има потребу да врши груписање торки према заједничким вредностима у одређеним атрибутима. За груписање се користе различите технике. На пример, за спајање:

- 1 Груба сила (енг. brute force) у којој се праве све могуће комбинације торки у спајању
- 2 Помоћу индекса који се користи за директан приступ упареним торкама унутрашње релације спајања
- 3 Помоћу хеша који се користи уместо индекса за директан приступ упареним торкама унутрашње релације спајања



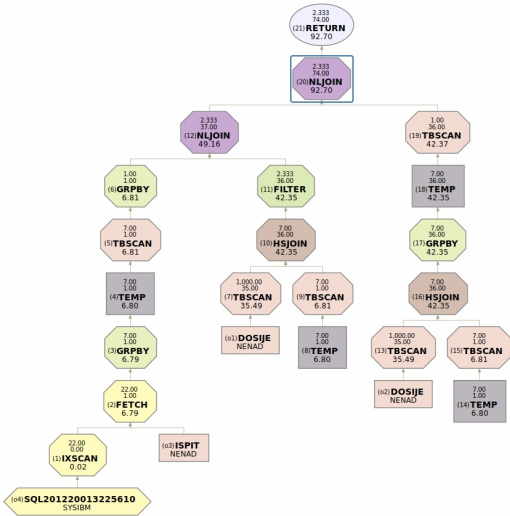
# Оптимизација у ДБ2

SQL упит у DB2 је скуп select-from-where блокова

- 1 Бира се редослед блокова, при чему се у случају угнеждених блокова оптимизује прво који је на највећој дубини (нај-унутрашњији блок)
- 2 EXPLAIN – алат за процену цене упита
- 3 Корисник DB2 LUW на располагању има Visual Explain – алат за процену са графичким интерфејсом
- 4 Потребно је извршити иницијализацију Explain табела преко Data Studio-а или са командне линије (директоријум MISC, датотека EXPLAIN.DDL



# Оптимизација у Д62



# Нивои оптимизације у Db2

## Нивои оптимизације у 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

# Увод

- Исти задатак може да се реши на више начина
- Решења могу да се разликују по ефикасности - како изабрати најефикасније
- 16.primeri.sql - различита решења истих задатака (два примера)
- Проценити ефикасност са Visual explain - који су разлози разлика?

# Увод

- *SARGable* - *Search ARGument* атрибути - атрибути по којима може да се врши претраживање
- Неке од наведених правила оптимизатор може да трансформише - видети Висуал Ехплаин
- Правила за превођење предиката су приказана у DB2 V11.5 Performance Tuning, Табела 61

# Ефикасност SELECT наредбе

- Навести само атрибуте који су неопходни - не користити "\*" ако нема потребе
- Користити предикате који праве рестрикцију само на оне случајеве који су потребни
- Ако је потребан значајно мањи број слогова броја постојећих у табели користити OPTIMIZE FOR клаузулу
- Користити FOR READ ONLY/FOR FETCH ONLY клаузуле
- Искључити DISTINCT/ORDER BY где нису неопходни
- Користити UNION ALL уместо UNION где је то могуће



# Ефикасност SELECT наредбе

- Сем у случају малих табела избегавати `SELECT count(*) from <tabela>` за проверу да ли је табела празна
- Користити `IN` листу ако се исти атрибут јавља у више предиката
- Ако је могуће, избећи коришћење `OR` предиката при спајању табела
- ...

# Нека правила за кодирање

Избегавати, уколико је могуће, коришћење скаларних функција над атрибутима у предикату

Уместо

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

ефикаснији запис је

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



# Нека правила за кодирање

Искључити, уколико је могуће, примену математички функција над атрибутима у предикату

Уместо

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

ефикаснији запис је

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

# Нека правила за кодирање

Искључити `DISTINCT` када год је то могуће. Ако треба елиминисати дупликате

- користити `GROUP BY` који може да користи индексе (ако постоје) ради елиминисања сортирања
- написати упит употребом `IN` или `EXISTS`. Корисно ако табела која враћа дупликате не враћа податке за неке вредности

# Нека правила за кодирање

Уместо

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

ефикаснији запис је

```
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
```

# Нека правила за кодирање

Уместо

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

ефикаснији запис је

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

# Нека правила за кодирање

Не тражити податке који су већ познати

Уместо

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

ефикаснији запис је

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

# Нека правила за кодирање

Користити CASE уместо UNION, ако је могуће

Уместо

```
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
```

# Нека правила за кодирање

ефикаснији запис је

```
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
```





# Нека правила за кодирање

- У кодирању дати предност *SARGable* атрибутима
- У кодирању обратити пажњу на конструкцију предиката над атрибутима где је дефинисан индекс
- Тип атрибута може се видеи у Visual Explain при одабиру процедуре ниског нивопа
- Детаљнији приказ у DB2 V11.5 Performance Tuning, Табеле 58 и 60
- *A Guide to Db2 Performance for Application Developers Code for Performance from the Beginning* - Craig S. Mullins
- Обратити пажњу на процедуре ниског нивоа - да ли могу да се промене



