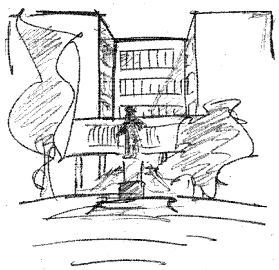
[P273]

Пројектовање база података

13а



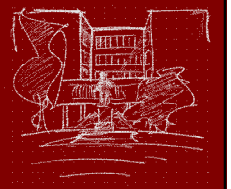
Саша Малков

Универзитет у Београду
Математички факултет

2023/2024

[P273]

Пројектовање база података



Саша Малков

Тема 16

Оптимизација упита

Основни методи оптимизације БП

- Оптимизација на нивоу интерне организације података
  - не уводи се разлика између физичког и логичког модела
  - остварује се кроз управљање интерном организацијом података, помоћним компонентама и ресурсима
    - управља се појединостима имплементације табела
- Оптимизација на нивоу упита
  - не уводи се разлика између физичког и логичког модела
  - врши се писање упита на начин који омогућава њихово ефикасније извршавање
  - овај вид оптимизације постепено губи на значају
    - савремени СУБП имају добру аутоматску оптимизацију
- Оптимизација на нивоу структуре података
  - ***уводи се*** разлика између физичког и логичког модела
  - организација табела се мења у односу на логички модел

Универзитет у Београду - Математички факултет

Опт. на нивоу интерне организације података

- Оптимизација на нивоу интерне организације података
  - Ова тема је углавном обрађена кроз упознавање физичке структуре података и прављење физичког модела базе података
  - Данас ћемо се бавити преосталим методима оптимизације

Универзитет у Београду - Математички факултет

Проблем оптимизације упита

- ***Оптимизација упита*** подразумева оптимизацију свих видова употребе базе података, што укључује и читање (упите у ужем смислу) и мењање садржаја базе података
- Важне теме везане за оптимизацију су:
  - начин извршавања упита
  - радно оптерећење базе података (енгл. *workload*)
- Оптимизација упита је зависна од конкретног СУБП
  - овде се представља на примеру система *DB2*

Начин извршавања упита

- Савремени СУБП извршава упита кроз неколико фаза:
  - прави се један или више планова извршавања упита
    - са процењеним трошковима извршавања
  - бира се најефикаснији план
    - и евентуално се додатно фино подешава
  - извршава се изабрани план
  - изабрани план може да се сачува
    - у зависности од имплементације и конфигурације СУБП

План извршавања упита

- План извршавања упита (енгл. *execution plan*) обухвата:
  - редослед корака при извршавању
  - операције које се извршавају у појединим корацима
  - структуре података које се употребљавају
  - начин сваког појединачног приступања подацима
  - процењену цену сваког од корака и целог посла
- Већина савремених СУБП омогућава кориснику да сагледа план извршавања пре него што се покрене израчунавање упита
  - можда не делује значајно за мање ад-хок упите, али је веома важно за упите који се понављају и скупе упите

Статички и динамички упити

- Према тренутку прављења плана извршавања упита, начин извршавања упита може да буде:
  - статички
    - план извршавања се прави унапред, при превођењу програма
    - програми са статичким упитима се зову *статички програми*
    - план извршавања упита неког програма се евидентира у БП
      - то се назива *уградњом* програма (енгл. *binding*)
  - динамички
    - план извршавања се прави при извршавању упита
    - програми који имају само динамичке упите називају се *динамички програми*
- Тренутак прављења плана извршавања производи последице, а навише у погледу
  - ауторизација
  - оптимизације

Статички и динамички упити (2)

- Последице у односу на ауторизацију:
  - корисник који уграђује програм мора да има ауторизацију за извршавање статичких упита и наредби
  - корисник који извршава програм мора да има ауторизацију за извршавање динамичких упита и наредби и ауторизацију за извршавање статичког програма
- На пример
  - ако статички програм извршава статички упит који рачуна просек плата
    - корисник који уграђује програм мора да има приступ појединачним подацима о платама
    - корисник који извршава програм мора да има ауторизацију да изврши програм
  - ако динамички програм извршава упит који рачуна просек плата
    - корисник који пише програм не мора да има никаква права приступа подацима
    - корисник који извршава програм мора да има приступ појединачним подацима о платама

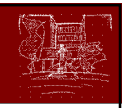
Статички и динамички упити (3)

- Последице у односу на ефикасност:
  - прављење плана извршавања није бесплатно
    - план извршавања статичких упита се прави само једанпут
    - план извршавања динамичких упита се прави сваки пут поново
      - савремени СУБП обично омогућавају неки вид кеширања планова
  - тренутак оптимизације може да буде значајан
    - оптимизација се изводи у тренутку прављења плана извршавања
    - динамички упити су оптимизовани на најбољи могући начин
    - статички упити су оптимизовани према стању БП у тренутку уградње пакета
      - то може бити неажурна оптимизација
      - зато постоји поступак *освежавања* планова (енгл. *package rebinding*)

Начин рачунања цене корака

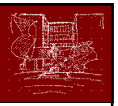
- При рачунању цене појединачне операције узимају се у обзир следеће информације
  - цена уређаја (приступање страници, читање странице)
  - број редова у табели
  - попуњеност страница
  - селективност упита (процена броја редова који задовољавају услов)
  - цена саме операције над дохваћеним подацима
- Већина тих информација је тачна само ако су **ажурни** статистички подаци о садржају базе података (!!!)

Врсте операција (пример DB2)



- Врсте операција ћемо навести на примеру система *DB2*
  - Није неопходно научити све операције напамет, али је важно разумети које операције постоје и чему служе
  - Посебну пажњу ћемо посветити операцијама које се најчешће користе

Врсте операција (пример DB2) (2)



- Најчешће операције:
  - TBSCAN – читање (претраживање) редова података непосредно из табеле
  - IXSCAN – претраживање индекса и читање показивача на редове
  - EISCAN – претраживање индекса по сложенијим условима и читање показивача на редове
  - RIDSCN – читање свих показивача на редове из индекса (или међурезултата)
  - FETCH – читање садржаја колона из датог реда (или редова)
  - HSJOIN – хеш спајање, без претходног уређивања
  - NLJOIN – угнеждено спајање, унутрашња табела се претражује за сваки ред спољашње
  - MSJOIN – *merge* спајање, када су обе табеле уређене по услову спајања
  - SORT – уређивање редова, опционо без понављања

Врсте операција (пример DB2) (3)

- Остале операције:
  - UNION – унија два скупа редова
  - UNIQUE – уклањања поновљених редова
  - GRPBY – груписање редова
  - RETURN – враћање резултата
  - DELETE – брисање једног или више редова
  - INSERT – додавање реда у табелу
  - UPDATE – ажурирање садржаја реда
  - GENROW – прављење нових редова података (трајних за INSERT или привремених)
  - TEMP – записивање података у помоћну привремену табелу

Врсте операција (пример DB2) (4)

- Остале операције:
  - IXAND – конјункција филтера по битмапираним индексима
  - FILTER – примена услова претраживања на скуп података
  - SHIP – читање података из удаљеног релационог или XML извора (нпр. федеративне базе)
  - RPD – читање података из удаљеног нерелационог извора (нпр. федеративне базе)
  - TQ – *ред табеле*, за преношење података између различитих агената (процеса)
  - XSCAN, XISCAN, XANDOR – израчунавање делова *Xpath* упита над *XML* фрагментима
  - CMPEXP , PIPE – помоћне операције, само за дебаговање

Неопходне операције

- Неке операције *морају* да се изврше да би се упит извршио
  - Такве операције се представљају ***неопходне операције***
  - Оне не могу да се оптимизују
- На пример, неопходне операције су:
  - RETURN
    - ако је у питању читање
  - DELETE / INSERT / UPDATE
    - ако је у питању нека од тих операција ажурирања
  - GRPBY
  - HSJOIN / NLJOIN / MSJOIN
  - UNION
- У неким случајевима су неопходне и:
  - SORT
  - UNIQUE

DB2 – План извршавања упита

- План извршавања упита може да се анализира помоћу различитих алата
  - Командни режим
    - Употребом тзв. *EXPLAIN* табела
    - Није посебно удобно и практично, више је намењено за аутоматизацију
  - *IBM Data Studio*
    - Направи се нови упит
    - Изабере се наредба *Open Visual Explain*
    - Може да се отвори у веб-прегледачу (*Open in New Window*)

Прављење помоћних табела

- При анализи и (мануелној) оптимизацији упита користи се скуп табела у истој бази у схеми *EXPLAIN*
  - Обично се називају *EXPLAIN*-табеле
  - Морају да се направе да би могла да се ради анализа упита
- Један од начина да се направе је:
  - db2 call sysproc.sysinstallobjects('EXPLAIN', 'C',
    cast(null as varchar(128)), cast(null as varchar(128)))

Примери упита

- Размотрићемо више врста једноставних и умерено сложених упита над базом података са подацима о студентима
  - Користићемо алат *Data Studio*
  - Користићемо базу података *Stud2020*

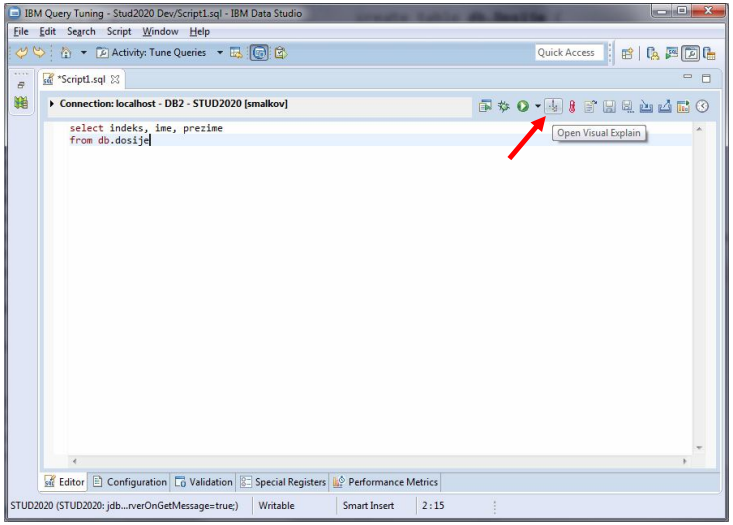
Пример базе података

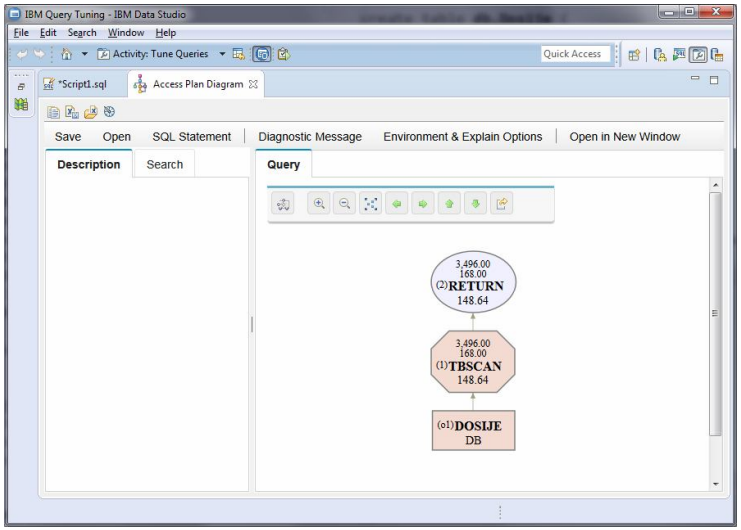
```
CREATE TABLE DB.DOSIJE  (
    INDEKS INTEGER NOT NULL,
    ID_SMERA INTEGER NOT NULL,
    STATUS VARCHAR(20) NOT NULL,
    IME VARCHAR(20) NOT NULL,
    PREZIME VARCHAR(25) NOT NULL,
    ...
    PRIMARY KEY (INDEKS)
)
```

```
CREATE TABLE DB.SMER (
    ID INTEGER NOT NULL,
    OZNAKA VARCHAR(8) NOT NULL,
    NAZIV VARCHAR(50) NOT NULL,
    SEMESTARA SMALLINT NOT NULL WITH DEFAULT 8,
    BODOVI SMALLINT NOT NULL WITH DEFAULT 180,
    ID_NIVOA SMALLINT NOT NULL,
    ZVANJE VARCHAR(40) NOT NULL WITH DEFAULT,
    OPIS LONG VARCHAR,
    PRIMARY KEY (ID)
)
```

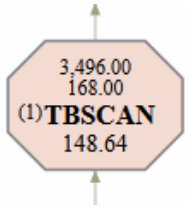
Анализа плана извршавања

- За анализирање најбољег процењеног плана извршавања користи се алат *Visual Explain* у оквиру алата *Data Studio*
  - Најпре се повежемо на базу података и направимо нови упит (*New SQL Scipt*)
  - Упишемо текст упита
- Размотрићемо више врста једноставних и умерено сложених упита над базом података са подацима о студентима
- Најпре ћемо илустровати поступак у алату *Data Studio*





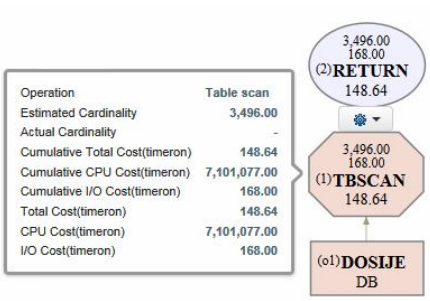
Елементи описа корака



- У опису сваког корака су следећи елементи (одозго наниже):
  - процењен број редова који ће се добити у резултату
  - кумулативна цена У/И операција
    - цена ове операције додата на цене припреме улаза
  - редни број (ознака) и назив операције
  - укупна кумулативна цена

- **ВАЖНО**: цене обраде (*CPU*), улаза и излаза (*I/O*) и укупна цена (*Total*) користе различите мерне јединице, иако су им називи исти, па зато може да изгледа да је укупна цена мања од цене улаза и излаза
- 1 У/И "тимерон" је цена читања једне странице

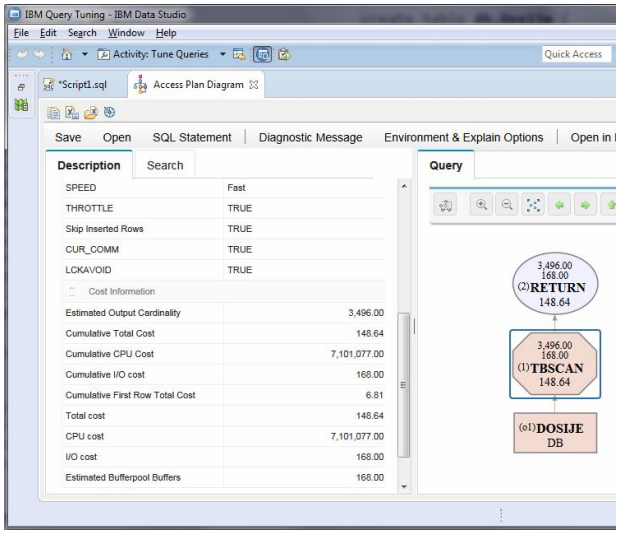
Елементи описа корака (2)



- Довођењем миша изнад корака добија се више елемената:
  - одвојено цена обраде и цена улаза и излаза
  - одвојено цена корака и кумулативна цена

* **ВАЖНО**: цене обраде (*CPU*), улаза и излаза (*I/O*) и укупна цена (*Total*) користе различите мерне јединице, иако су им називи исти, па зато може да изгледа да је укупна цена мања од цене улаза и излаза

Елементи описа корака (3)



- Још више елемената описа корака се добија ако се кликне на корак
  - У левом делу прозора се отвара листа различитих описа корака

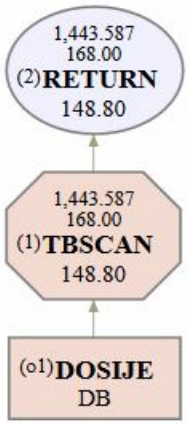
Упит са рестрикцијом

- У наредном кораку проценићемо трошкове извршавања упита са рестрикцијом:

```
select indeks, ime, prezime
from db.dosije
where idprograma = 101
```

- Процена цене зависиће од неколико фактора:
  - Да ли постоји индекс који одговара услову рестрикције?
    - Ако постоји, онда можда његова употреба може да допринесе?
  - Колика је разноликост садржаја колоне (или колона) по којима се врши рестрикција?
    - Ажурни статистички подаци могу да помогну у процењивању цене.

Упит са рестрикцијом, без индекса




```
select indeks, ime, prezime
from db.dosije
where idprograma = 101
```

- Приметимо да је цена улазно/излазних операција остала иста
  - и даље је неопходно прочитати све странице табеле
- Процењен број редова резултата је мањи него пре, зато што се издвајају само неки редови

Упит са рестрикцијом са индексом

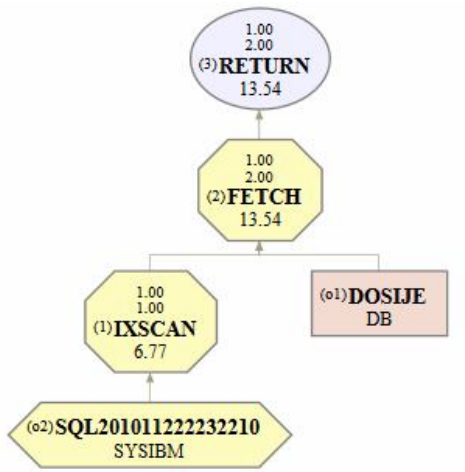
- У наредном кораку проценићемо трошкове извршавања упита са рестрикцијом по атрибуту за који постоји направљен индекс:

```
select indeks, ime, prezime
from db.dosije
where indeks = 20150201
```



```
select indeks, ime, prezime
from db.dosije
where indeks = 20150201
```

- Операција *IXSCAN* издваја идентификаторе редова (*RID*) који одговарају вредности индекса
- Операција *FETCH* на основу списка *RID*-ова (у овом случају тачно један ред) чита дате редове из табеле



Упит са рестрикцијом са индексом (2)

- Ако направимо индекс по студијском програму:

```
create index db.Dosije_StProgram
on db.Dosije( IdPrograma )
```

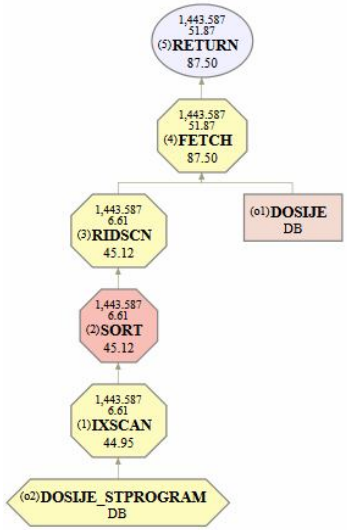
- онда ће рестрикција бити по атрибуту који је индексиран, али који има понављања:

```
select indeks, ime, prezime
from db.dosije
where idprograma = 101
```



```
select indeks, ime, prezime
from db.dosije
where idprograma = 101
```

- Операција *SORT* уређује прочитане идентификаторе редова, да би касније читање ишло редом
  - цена читања је 0
  - плаћа се само уређивање у меморији
- Операција *RIDSCAN* чита *RID*-ове из претходног резултата
  - цена је скоро нула



Упит са спајањем

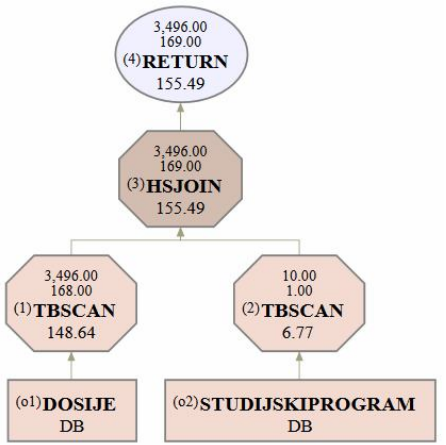
- У наредном примеру ћемо да спајамо податке о студентима са подацима о смеровима:

```
select ime, prezime, sp.naziv
from db.dosije d
 join db.StudijskiProgram sp
  on d.IdPrograma = sp.Id
```



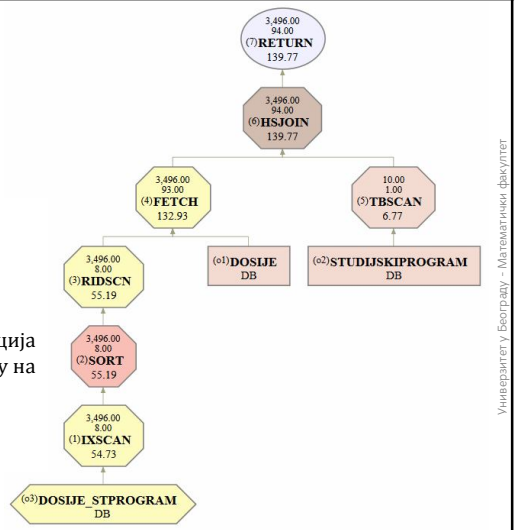
```
select ime, prezime, sp.naziv
from db.dosije d
 join db.StudijskiProgram sp
  on d.IdPrograma = sp.Id
```

- Најпре без индекса
  - чита се све из табеле *DOSIJE*
- Операција *HSJOIN:*
  - најпре за бар једну од колекција прави у меморији хеш табелу на основу услова спајања
    - не уређује редове
  - затим спаја податке




```
select ime, prezime, sp.naziv
from db.dosije d
 join db.StudijskiProgram sp
  on d.IdPrograma = sp.Id
```

- Ако постоји индекс
  - чита се из табеле *DOSIJE*
  - али већ уређено
- Операција *HSJOIN:*
  - најпре за бар једну од колекција прави у меморији хеш табелу на основу услова спајања
    - не уређује редове
  - затим спаја податке



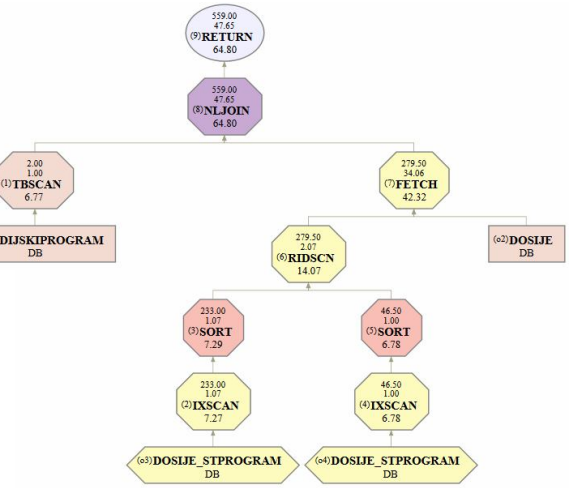
Упит са спајањем и рестрикцијом

- Спајање података о студентима са подацима о смеровима, као и малопре, али сада са рестрикцијом по смеру:

```
select ime, prezime, sp.naziv
from db.dosije d
 join db.StudijskiProgram sp
  on d.IdPrograma = sp.Id
where sp.id in (201, 301)
```

План извршавања упита, примери

```
select ime, prezime, sp.naziv
from db.dosije d
  join db.StudijskiProgram sp
    on d.IdPrograma = sp.Id
where sp.id in (201, 301)
```



- Читају се сви редови из табеле *STUDIJSKIPROGRAM*
- Прочитају се само тражени редови табеле *DOSIJE*
- Затим се врши спајање оператором *HSJOIN*

Мало сложенији примери...

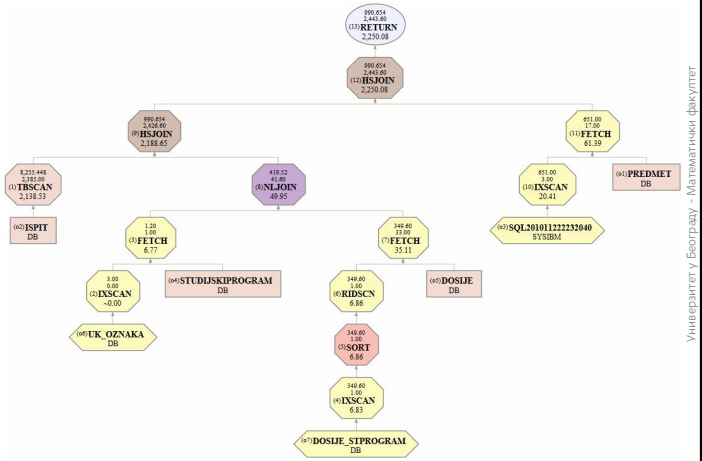
Сви испити положени у школској 2018/19. години на смеру Информатика (основне студије)

```
select d.ime, d.prezime, p.naziv, ip.ocena
from db.dosije d
  join db.studijskiprogram s
    on d.idprograma = s.id
    and s.naziv='Информатика'
    and s.idnivoa = 1
  join db.ispit ip
    on ip.indeks = d.indeks
    and ip.ocena > 5
    and ip.skgodina = 2018
  join db.predmet p
    on ip.idpredmeta = p.id
```


Мало сложенији примери...

Сви испити положени у школској 2018/19. години на смеру Информатика (основне студије)

```
select d.ime, d.prezime,
            p.naziv, ip.ocena
from db.dosije d
  join db.studijskiprogram s
    on d.idprograma = s.id
    and
s.naziv='Информатика'
    and s.idnivoa = 1
  join db.ispit ip
    on ip.indeks = d.indeks
    and ip.ocena > 5
    and ip.skgodina = 2018
  join db.predmet p
    on ip.idpredmeta = p.id
```



Некада се индекси не користе како бисмо претпоставили...

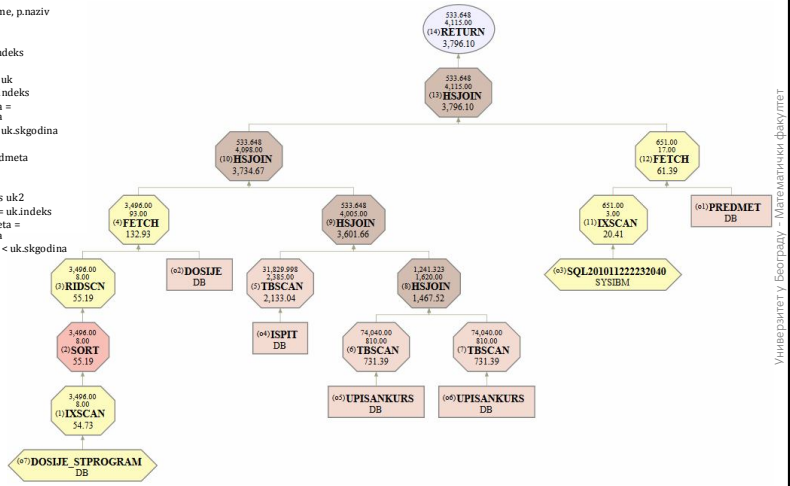
Студенти који су положили испит после поновљеног слушања предмета

```
select d.ime, d.prezime, p.naziv
from db.dosije d
  join db.ispit ip
    on ip.indeks = d.indeks
    and ip.ocena > 5
  join db.upisankurs uk
    on ip.indeks = uk.indeks
    and ip.idpredmeta = uk.idpredmeta
    and ip.skgodina = uk.skgodina
  join db.predmet p
    on p.id = uk.idpredmeta
where exists (
  select *
  from db.upisankurs uk2
  where uk2.indeks = uk.indeks
    and uk2.idpredmeta = uk.idpredmeta
    and uk2.skgodina < uk.skgodina
)
```

Некада се индекси не користе како бисмо претпоставили...

Студенти који су положили испит после поновљеног слушања предмета

```
select d.ime, d.prezime, p.naziv
from db.dosije d
  join db.ispit ip
    on ip.indeks = d.indeks
    and ip.ocena > 5
  join db.upisankurs uk
    on ip.indeks = uk.indeks
    and ip.idpredmeta =
        uk.idpredmeta
    and ip.skgodina = uk.skgodina
  join db.predmet p
    on p.id = uk.idpredmeta
where exists (
  select *
  from db.upisankurs uk2
  where uk2.indeks = uk.indeks
    and uk2.idpredmeta =
        uk.idpredmeta
    and uk2.skgodina < uk.skgodina
)
```

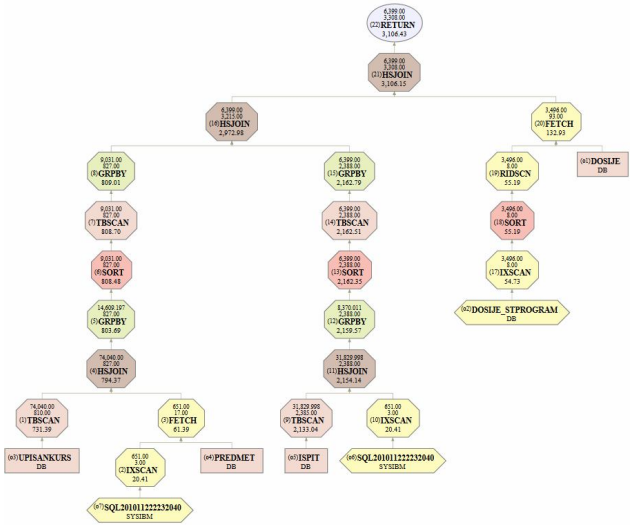


Сложеније операције са груписањем...

Преглед уписивања и полагања испита

```
with upisano as (
  select indeks, skgodina, count(*) n, sum(espb) espb
  from db.upisankurs uk
    join db.predmet p
      on uk.idpredmeta = p.id
  group by indeks, skgodina
),
polozeno as (
  select indeks, skgodina, count(*) n, avg(ocena) ocena
  from db.ispit ip
    join db.predmet p
      on ip.idpredmeta = p.id
  where ocena > 5
  group by indeks, skgodina
)
select d.indeks, ime, prezime, u.skgodina,
       u.n n_upis, u.espb, p.n n_polozeno, p.ocena
from db.dosije d
  join upisano u
    on d.indeks = u.indeks
  join polozeno p
    on p.indeks = d.indeks
    and p.skgodina = u.skgodina
```


Сложеније операције са груписањем...



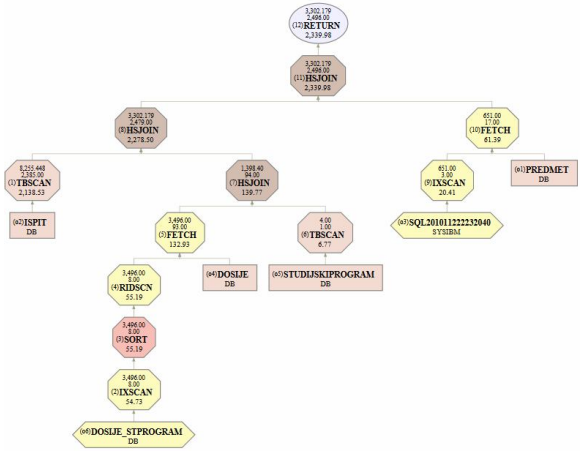
Оптимизација упита

- Оптимизација упита је мануелно или аутоматско одабирање најповољнијег плана извршавања упита ради постизања бољих перформанси
- Већина савремених СУБП располаже солидним оптимизаторима упита
- Ради илустрације, у наредна три примера се користе различити а еквивалентни упити, за које СУБП рачуна исте планове извршавања...

Верзија 1

Сви испити положени у школској 2018/19. години на смеру Информатика

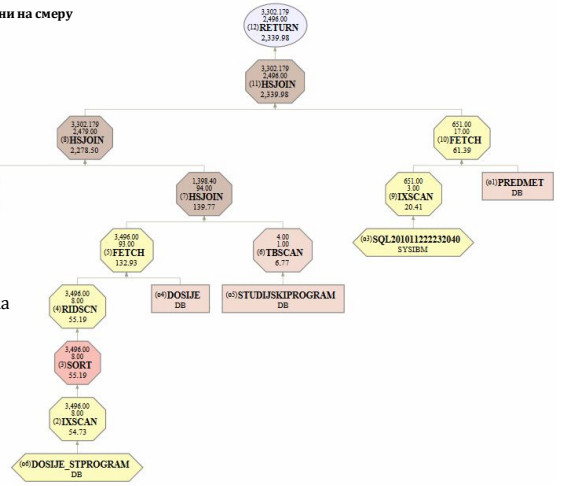
```
select d.ime, d.prezime,
    p.naziv, ip.ocena
from db.dosije d
  join db.studijskiprogram s
    on d.idprograma = s.id
     and s.naziv='Информатика'
  join db.ispit ip
    on ip.indeks = d.indeks
    and ip.ocena > 5
    and ip.skgodina = 2018
  join db.predmet p
    on ip.idpredmeta = p.id
```



Верзија 2

Сви испити положени у школској 2018/19. години на смеру Информатика

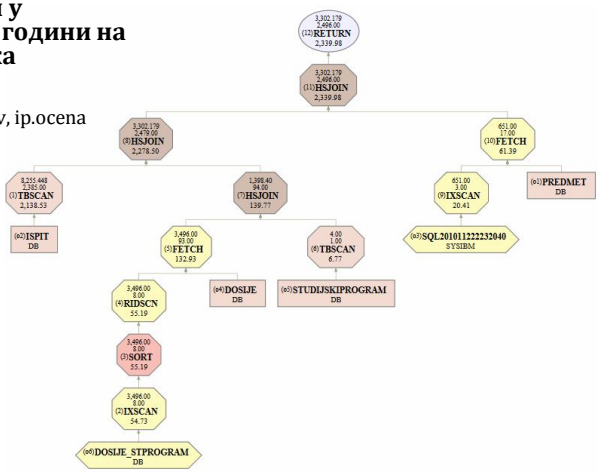
```
with ip as (
  select indeks, idpredmeta, ocena
  from db.ispit
  where ocena > 5
    and skgodina = 2018
),
smerovi as (
  select id
  from db.studijskiprogram
  where naziv='Информатика'
)
select d.ime, d.prezime, p.naziv, ip.ocena
from ip
  join db.dosije d
    on d.indeks = ip.indeks
  join db.predmet p
    on ip.idpredmeta = p.id
where d.idprograma in (
  select * from smerovi
)
```



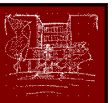
Верзија 3

Сви испити положени у школској 2018/19. години на смеру Информатика

```
select d.ime, d.prezime, p.naziv, ip.ocena
from db.dosije d,
   db.studijskiprogram s,
   db.ispit ip,
   db.predmet p
where
   d.idprograma = s.id
   and s.naziv='Информатика'
   and ip.indeks = d.indeks
   and ip.ocena > 5
   and ip.skgodina = 2018
   and ip.idpredmeta = p.id
```



Мануелна оптимизација упита



- Мануелна оптимизација упита је раније била много потребнија него данас
  - пројектант или администратор може на различите начине да сугерише СУБП-у који план извршавања да примени
- Данас је улога мануелне *оптимизације* упита важнија у фази прављења физичког модела него у експлоатацији
  - не оптимизује се упит, него се анализирају планови извршавања ради сагледавања и превазилажења евентуалних слабости предложеног модела

Фактори одлучивања

- При проналажењу *најбољег* плана извршавања узимају се у обзир расположиви подаци о бази података:
  - структура табела и кључева
  - структура упита
  - постојећи индекси
  - статистички подаци о садржају табела и атрибута
  - подаци о брзини физичких уређаја
  - величина бафера страница

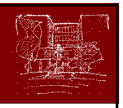
Статистике о бази података

- Да би процена трошкова извршавања била што тачнија, морају да постоје *исцрпне* и *ажурне* статистике о бази података
- Статистике морају да се праве на реалном садржају базе података
- Детаљност и прецизност статистика може да се бира
  - ако су детаљније и прецизније онда су
    - корисније и омогућавају ефикасније упите
    - обимније и скупље за чување и коришћење
    - сложеније и скупље за израчунавање

Статистике о бази података

- Статистике о табелама могу да се рачунају
  - за све редове или само изабране редове
    - редови се могу бирати на различите начине, уз одређивање степеном заступљености и начином изабирања узорка
  - за сваку од колона или само за изабране колоне
    - колоне примарног кључа, страних кључева, мануелно изабране,...
  - само опсег вредности или детаљна расподела
    - дистрибуција вредности (фреквенција, квантили и сл.)
  - аутоматски или мануелно
- Сличне статистике се праве и за индексе

DB2 – RUNSTATS



- Наредба *RUNSTATS* ажурира статистичке податке о датој табели или скупу табела:
  - RUNSTATS ON (TABLE | VIEW) <tabname> <options>
  - RUNSTATS ON TABLE DB.DOSIJE
    - ажурира податке о табели DB.DOSIJE
  - RUNSTATS ON TABLE DB.DOSIJE FOR INDEXES
    - ажурира податке о индексима табеле DB.DOSIJE
  - RUNSTATS ON TABLE DB.DOSIJE DETAILED FOR INDEXES
    - ажурира податке о индексима табеле DB.DOSIJE
- Велики број опција омогућава фино подешавање, као на пример:
  - одабир колона
  - начин одабира узорка редова за детаљну анализу
  - и друго...

DB2 – REORGCHK

- Наредба *REORGCHK* користи постојеће или ажурира статистичке податке за једну или више табела и процењује да ли је потребно да се изврши физичка реорганизација редова
  - REORGCHK [(UPDATE | CURRENT) STATISTICS] ON (SCHEMA ... | TABLE ...)
  - REORGCHK UPDATE STATISTICS ON TABLE ALL
    - ажурира статистичке податке и даје извештај о потреби за реорганизацијом за све табеле базе података

DB2 – REORGCHK (2)

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                      CARD      OV      NP      FP ACTBLK    TSIZE  F1  F2  F3
   REORG
-------------------------------------------------------------------------------------------
Table: DB.DOSIJE
                                 4319       0     393     393      -  1515969   0  98 100 ---
Table: DB.ISPIT
                               118263       0    2509    2509      -  9934092   0  98 100 ---
...
```

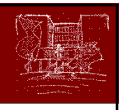

DB2 – REORGCHK (3)

```
Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (Space used on leaf pages / Space available on non-empty leaf pages) > MIN(50, (100 -
    PCTFREE))
F6: (100 - PCTFREE) * (Amount of space available in an index with one less level / Amount of space
    required for all keys) < 100
F7: 100 * (Number of pseudo-deleted RIDs / Total number of RIDs) < 20
F8: 100 * (Number of pseudo-empty leaf pages / Total number of leaf pages) < 20

SCHEMA.NAME                 INDCARD  LEAF ELEAF LVLS  NDEL    KEYS LEAF_RECSIZE NLEAF_RECSIZE
    LEAF_PAGE_OVERHEAD NLEAF_PAGE_OVERHEAD  PCT_PAGES_SAVED  F4  F5  F6  F7  F8 REORG
-------------------------------------------------------------------------------------------------
    ------------------------------------------------------------------
Table: DB.DOSIJE
Index: DB.DOSIJE_IME
                             4319    85     0    3     0    4319           53            53
                  566                 566                0   3  91  46   0   0 *----
Index: SYSIBM.SQL151204073209750
                             4319    25     0    2     0    4319            4             4
                  710                 710                0  43  74   -   0   0 *----
...
```


...још о оптимизацији упита...



- Прочитати поглавља 12 и 13 из *Ramakrishnan, Gehrke,* ***Database Management Systems****, 2.ed, 2000.*

Оптимизације на нивоу структуре података

- Прелазимо на разматрање оптимизације на нивоу структуре података
- На овом нивоу оптимизације ***мења се*** физичка структура података у односу на логички модел

Оптимизације на нивоу структуре података

- Ако имамо узорак радног оптерећења и одговарајуће планове извршавања, онда можемо да спроведемо оптимизацију структуре базе података
  - у циљу повећавања перформанси датог радног оптерећења
- Оптимизација физичког дизајна мењањем физичке структуре базе података

Узорак радног оптерећења

- Узорак радног оптерећења (енгл. *workload*) је скуп типичних упита (и промена базе података) за које се зна (или се процењује) да ће чинити најчешћи облик приступања бази података
- Добар узорак се састоји од
  - скупа упита и промена базе података
  - за сваки упит и промену постоји и процењена учесталост извршавања и посебно вршно оптерећење
  - формулисаних циљних перформанси

На основу узорка...

- За упите:
  - препознаје се које се релације (табеле, индекси) користе
  - који атрибути се издвајају
  - који атрибути учествују у условима спајања и рестрикције
- За промене базе, поред тога:
  - врста промене
  - атрибути који се мењају (за ажурирање)

Циљ структурне оптимизације

- Препознавање индекса које морамо да направимо
  - ...о томе је већ било речи, али биће још нешто касније...
- Препознавање потребних промена у физичкој схеми
  - Увођење погледа да би се сакриле начињене промене
  - Алтернативна нормализација
  - Денормализација

Увођење погледа да би се сакриле начињене промене

- Ако се промени физичка структура базе података, онда на концептуалном нивоу могу да се уведу нови погледи који скривају начињене измене
- Ово није део физичког модела, али је последица промена у физичком моделу

Алтернативна нормализација

- При нормализовању не постоји само једно могуће решење
  - нормализација до 3.НФ или БКНФ
  - нормализација до 3.НФ различитим путевима
- Ако једно решење не даје задовољавајуће резултате, можда друго даје?

Пример различитих нормализација

- Нека релација *P* описује пројекте:
  - *P(BrProjekta, Grad, Naziv)*
- Имамо следеће ф.зависности:
  - fd1: *BrProjekta → Grad*
  - fd2: *BrProjekta → Naziv*
- Ако у једном граду постоји највише један пројекат и сви пројекти имају различите називе, онда можемо да сматрамо да постоји и:
  - fd3: *Grad → Naziv*
  - fd4: *Naziv → Grad*
  - fd5: *Naziv → BrProjekta*
- Одатле следи да имамо и алтернативне нормализације:
  - Нормализација 1:
    - *P(BrProjekta, Grad)*
    - *PN(Grad, Naziv)*
  - Нормализација 2:
    - *P(BrProjekta, Naziv)*
    - *PG(Naziv, Grad)*
  - Па чак и Нормализација 3:
    - *P(BrProjekta, Grad)*
    - *PN(BrProjekta, Naziv)*

Денормализација

- *Денормализација* је промена структуре БП тако да се свесно нарушава достигнута нормална форма
- Оптимизација структуре базе података често представља вид денормализације
- Неке технике
  - подела табеле (декомпозиција)
  - спајање табела (композиција)
  - дуплирање података

Употреба *слабије* НФ

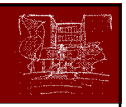
- Јача НФ може да доследније поштује функционалне зависности, али да уведе вишу цену неких операција
  - пример на једној од претх. страна
- Међутим, ако употребимо слабију НФ и додатне услове интегритета, то може да буде још скупље:

```
P( BrProjekta, Grad, Naziv )
Check( not exists (
            select naziv from P
            group by naziv
            having count(distinct grad)>1
    ))...
```


Употреба *слабије* НФ

- Свако слабљење нормализације захтева да се функционалне зависности одржавају на неки други начин
  - додатни услови интегритета...
- Морамо да проверимо да ли се оно што денормализацијом добијамо при рачунању упита, са друге стране губи при мењању садржаја базе података

Подела табеле (декомпозиција)



- Ако имамо табелу са много редова или колона, која се често мења, онда промене могу да буду неефикасне
  - индекси постају „дубоки“ и често се чита са диска
  - упити који не користе индексе су тим пре неефикасни
- Један начин да се рад убрза је да се табела подели на две или више табела
- Имамо две врсте поделе: хоризонталну и вертикалну

Хоризонтална подела табеле

- Редови табеле се поделе у две табеле (или више табела) са истом структуром
  - Обично једна садржи тзв. нове или активне податке, а друга тзв. старе или архивске податке
  - Или се подаци групишу према другом критеријуму
    - начину употребе (на пример, по студијском програму)
    - врсти података (на пример, по нивоу студија)
    - ...
- Добре стране:
  - убрзавају се неке операције (пре свега додавање и мењање нових података)
- Лоше стране:
  - додатно се компликују неке друге операције, пре свега аналитички упити над свим подацима

Партиционисање табела

- Вид хоризонталне поделе је и партиционисање табела
- Често може да почива на неким критеријумима распона вредности
  - на пример, подаци за сваку школску годину могу да иду у посебну партицију
- Што је више делова
  - то су појединачне операције са подацима ефикасније,
  - али је претраживање целе табеле мање ефикасно

Вертикална подела табеле

- Ако се већина колона не користи у свим упитима, већ само релативно ретко, онда се честе операције могу убрзати вертикалном поделом
- Праве се две или више табела:
  - свака садржи колоне примарног кључа
  - све остале колоне се групишу према употреби
- Добре стране:
  - убрзавају се неке честе операције
- Лоше стране:
  - додатно се компликују неке операције, пре свега аналитички упити над свим подацима
- (ово није денормализација, напротив, али се ова врста промена обично разматра заједно са денормализацијама)

Спајање табела

- Табеле које се често (или чак сваки пут) спајају у упитима, може да буде корисно спојити у једну табелу
- Добре стране:
  - Спајање је веома скупа операција и оваква промена може да има значајне ефекте
- Лоше стране:
  - Добијена табела обично нарушава нормалне форме и садржи неке редундантности
  - То додатно отежава одржавање података и старање о интегритету
  - Може да се значајно повећа број редова или укупно заузеће простора

Понављање података

- Једна од техника оптимизације може да буде да се неки подаци свесно понове у више табела
- Добре стране:
  - Избегавају се сувишна спајања или уније
- Лоше стране:
  - Редундантност, тј. отежано одржавање и повећано заузеће простора
    - потенцијално потребни додатни услови интегритета

Друге врсте денормализације

- Већ поменуте промене спајања табела и понављања података представљају видове денормализације
  - Други облици денормализације су различити видови премештања или копирања колона из једне табеле у другу, а који за циљ имају подизање перформанси (примарно кроз смањивање броја операција спајања)
  - Сви видови денормализације имају за последицу увођење редундантности и све што из тога следи
- Примери:
  - два ентитета у односу 1-1 могу да се моделирају једном табелом
  - два ентитета у односу 1-Н могу да се моделирају једном табелом

Догма о денормализацији

- Уврежен је став да је нормализована база података неефикасна
- Међутим, то важи само за специфичне околности и није опште правило
- Денормализација врло често доприноси ефикасности или сасвим мало или нимало
- То је само један од могућих корака

Догма о денормализацији (2)

- Утицај декомпоновања на перформансе је веома разноврстан
  - увећање ефикасности ажурирања
    - због елиминације редундантности
  - увећање ефикасности читања
    - смањењем редундантности се смањује укупан обим података
  - смањење ефикасности ажурирања
    - зато што се ажурира више табела (и индекса)
  - смањење ефикасности читања
    - зато што морају да се спајају табеле
- Некада је лако тачно проценити и предвидети последице, али некада није

Догма о денормализацији (3)

- На ефикасност неке конкретне структуре базе података утичу сложени фактори и посебно њихова комбинација:
  - нормализованост
  - обим података у различитим табелама
  - заступљеност операција читања и писања
  - сложеност операција читања и писања
- Који фактор има значајнији утицај и колико која промена структуре података доприноси перформансама често мора да се проверава експериментално

Разрешавање хијерархија

- Хијерархије ентитета смо помињали у моделу ентитета и односа
- Хијерархије постоје и у објектном моделу и дијаграму класа података
- При прављењу логичког модела хијерархија може да се преведе у релације (табеле) на бар три основна начина:
  - једна интегрална табелама
  - за сваки тип засебна табела, која има само колоне кључа и колоне нових атрибута
  - за сваки неапстрактан тип засебна табела, која има све потребне колоне
  - различите комбинације

Разрешавање хијерархија (2)

- Чест проблем са хијерархијама је да избор логичког модела који је “чистији” може да буде “спорији”
  - при прављењу логичког модела се разматрају критеријуми „логичких веза“ међу подацима
  - при прављењу физичког модела због ефикасности може да буде боље да се примени друго решење

Стабилно решење за хијерархије

- Пожељно је да се спречи да касније оптимизације физичког модела промене начин употребе табела
- То може да се оствари апстраховањем:
  - Да се све операције читања одвијају кроз погледе који заокружују све потребне податке о елементима хијерархије
  - Да се „забране“ директне операције мењања података у табелама које припадају хијерархији
  - Да се сва писања изводе кроз окидаче на погледима или предефинисане процедуре

Погледи са окидачима

- Један од проблема са оптимизацијама структуре БП је да се добија имплементација која одступа од логичког модела
- Специфичности физичког модела могу да се сакрију погледима
  - исто као што се погледима могу сакрити и спефцифичности логичког модела у односу на концептуални
- Проблем класичних погледа је да се иоле сложенији погледи не могу користити за мењање података, већ само за читање
- То се решава помоћу погледа са окидачима

Погледи са окидачима (2)

- Основни проблем са сложеним погледима је што из њихове дефиниције СУБП не може да једнозначно протумачи како би се имплементирале операције које мењају податке “у погледу”
- *Окидачи на погледима* представљају средство да се формално опише како се те операције изводе
- Окидачи на погледима се дефинишу као *замена* за операције непосредног мењања података

Погледи са окидачима (3)

- Наредба прављења окидача има облик:

```
CREATE [OR REPLACE] TRIGGER <ime>
    INSTEAD OF { INSERT | UPDATE [OF ...] | DELETE }
    ON <ime pogleda>
    ...
```


Литература за тему

- Гордана Павловић-Лажетић, **Увод у релационе базе података**, *2.изд. Математички факултет, 1999.*
  - доступно онлајн: *http://poincare.matf.bg.ac.rs/~gordana/urbp-2016.htm*
- Teorey, et.al, **Database Design, know it all,** 2008.
- Ramakrishnan, Gehrke, **Database Management Systems**, *2.ed, 2000.*
- Документација за *DB2 11.5*:
  - Онлајн:
    - https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0
  - ПДФ:
    - https://www.ibm.com/support/pages/node/627743