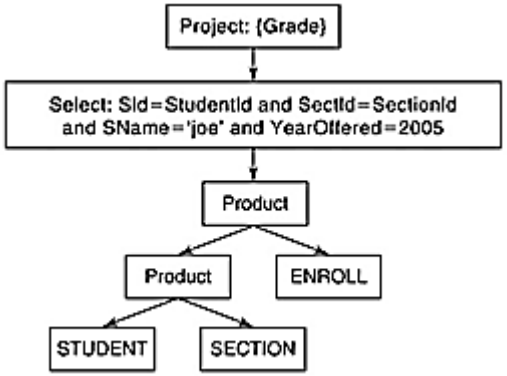
VT Sistem Gerçeklemesi Ders Notları- #14

SORGU OPTİMİZASYONU

- Sorgu Optimizasyonu ihtiyacı
- Ağaç dönüşümleri
- Ağaç kestirimi
  - En iyi ağaç Kestirimi için Sezgiler
  - JOIN sırası için Sezgiler
  - Dinamik Programlama ile Kapsamlı sayım
- Plan Kestirimi
- SimpleDB Optimizasyonu

Sorgu Optimizasyonu ihtiyacı

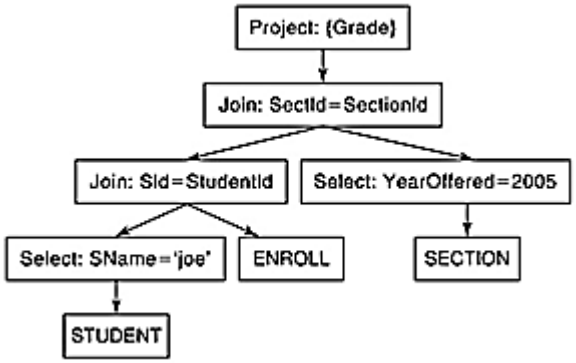


B(P1)= 4500+45000*2500 = 112504500

R(P1)=45000*25000 = $1125*10^6$

B(P2) = 112504500 + $1125*10^6*50000$ =**<u>56 trilyon</u>**

R(P2)= $1125*10^6$ * $1.5*10^6$



B(Select1)= 4500, R(Select1)= 1

B(J1) = 4500+ 1* 50.000 = 54500, R(J1) = 1.5milyon/45.000 =34

B(Select2) = 2500, R(Select2)= 500

B(J2) = 54500 + 34*2500 = **<u>139500</u>**, R(J2)= 34*500 / max(500, 34) = 34

k=50-blok

B(p8)= 2500+50+50+4534= **<u>7134</u>**

STUDENT	4,500	45,000	45,000 44,960 50 40	for F=SId for F=SName for F=GradYear for F=MajorId
SECTION	2,500	25,000	25,000 500 250 50	for F=SectId for F=CourseId for F=Prof for F=YearOffered
ENROLL	50,000	1,500,000	1,500,000 25,000 45,000 14	for F=EId for F=SectionId for F=StudentId for F=Grade

```
SimpleDB.init("studentdb");
Transaction tx = new Transaction();

// the plan for the STUDENT node
Plan p1 = new TablePlan("student", tx);

// the plan for the select node above STUDENT
Predicate joepred = new Predicate(...); //sname='joe'
Plan p2 = new SelectPlan(p1, joepred);

// the plan for the ENROLL node
Plan p3 = new TablePlan("enroll", tx);

// an indexjoin plan between STUDENT and ENROLL
MetadataMgr mdMgr = SimpleDB.mdMgr();
Map<String,IndexInfo> indexes =
                        mdMgr.getIndexInfo("enroll", tx);
IndexInfo ii = indexes.get("studentid");
Plan p4 = new IndexJoinPlan(p2, p3, ii, "sid", tx);

// the plan for the SECTION node
Plan p5 = new TablePlan("section", tx);
```

Figure 24-12
An efficient plan for the tree of Figure 24-11(c)

B(p4)= 4500+ 34=4534
R(p4)=34

B(p6)= 2500
R(p6)=500

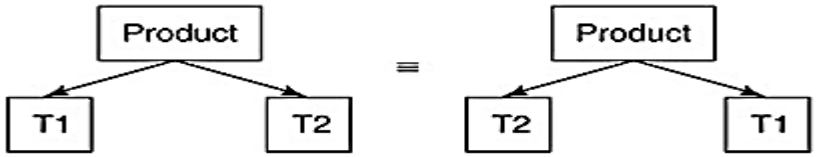
```
// the plan for the select node above SECTION
Predicate sectpred = new Predicate(...); //yearoffered=2005
Plan p6 = new SelectPlan(p5, sectpred, tx);

// use a multibuffer product plan to join SECTION
Plan p7 = new MultiBufferProductPlan(p4, p6, tx);
Predicate joinpred = new Predicate(...); //sectid=sectionid
Plan p8 = new SelectPlan(p7, joinpred, tx);

// the plan for the project node
List<String> fields = Arrays.asList("grade");
Plan p9 = new ProjectPlan(p8, fields, tx);
```

Figure 24-12 *(Continued)*

Sorgu Optimizasyonu adımları

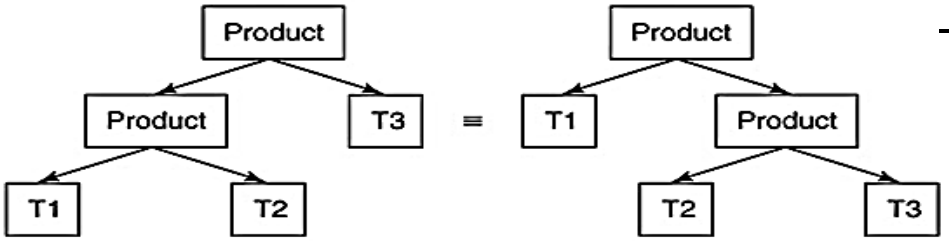
- N PRODUCT(JOIN)
  - N+1 tablo ➔ (2N)!/N!
  - 10 tablo➔ 9 product ➔ 176 milyar farklı sorgu ağacı
- Diğer düğümler:
  - *Select,Project,Materilize, Order, Group by, ..*
- Farklı düğüm gerceklemeleri
  - IndexSelect, IndexJoin, BNL, MergeJoin, Hash Join
- PLANLAYICI adımları:
  1. En iyi **AĞAC** KESTİRİMİ *(kayıt sayısı esas alınıyor..)*
  2. En iyi **PLAN** KESTİRİMİ *(blok erişim sayısı esas alınıyor..)*
- SEZGİ (*heuristic*): el yordamı ile ortaya atılan kesin olmayan kural

Ağaç Dönüşümleri-1: *product*



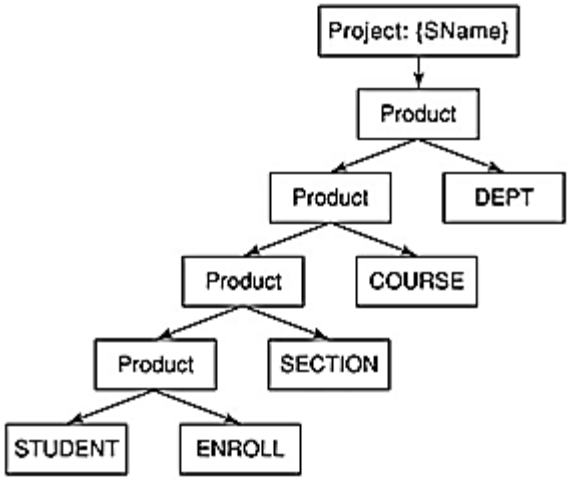
(a) The *product* operator is commutative

– Product(T1,T2) = product(T2,T1)

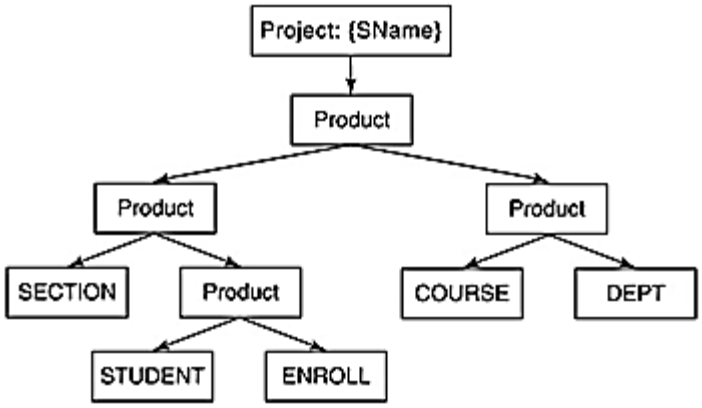


(b) The *product* operator is associative

– Product(product(T1,T2),T3) = product(T1,product(T2,T3))

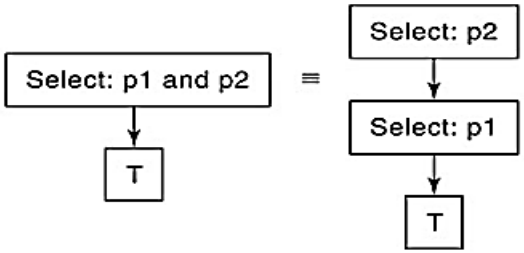


X(*X(S,E)*,S) ➔X(S,*X(S,E)*)
X(X(X(S,X(S,E)),C),D)➔
X(X(S,X(S,E)),X(C,D))

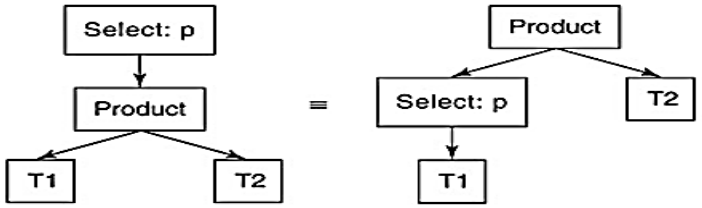


4

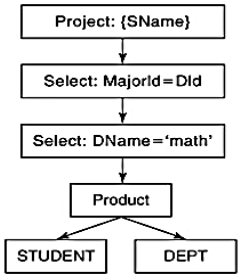
Ağaç Dönüşümleri-2: *select*

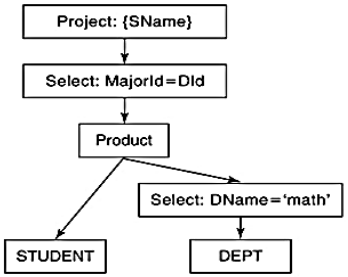


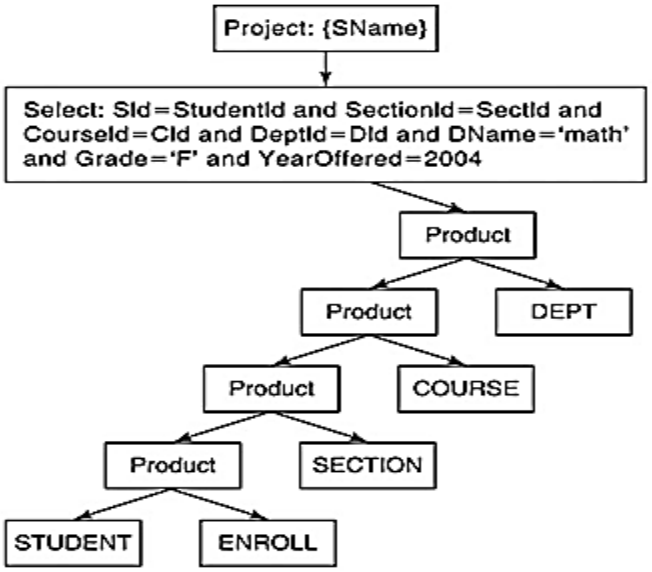
- SELECT (T, p1 and p2) = SELECT(SELECT(T,p1),p2)
- CNF(*conjuctive normal form*) yüklemi
  - ~~(MajorId=10 AND Sid=3) OR GradYear=2004~~
  - (MajorId=10 OR GradYear=2004) AND (Sid=3 OR GradYear=2004)

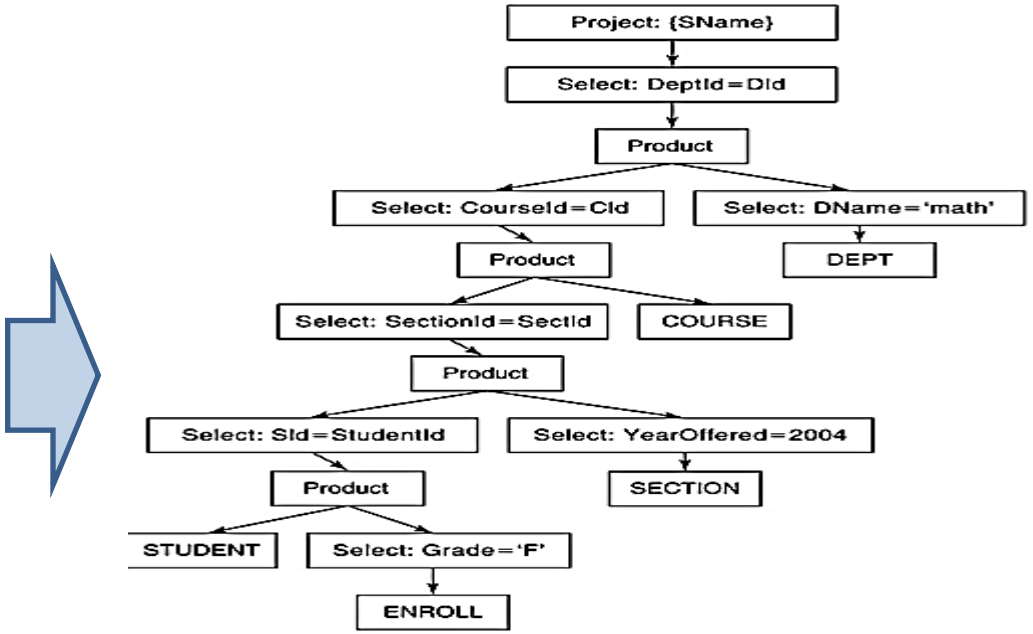


- SELECT (PRODUCT(T1,T2),p) = PRODUCT(SELECT(T1,p),p2)



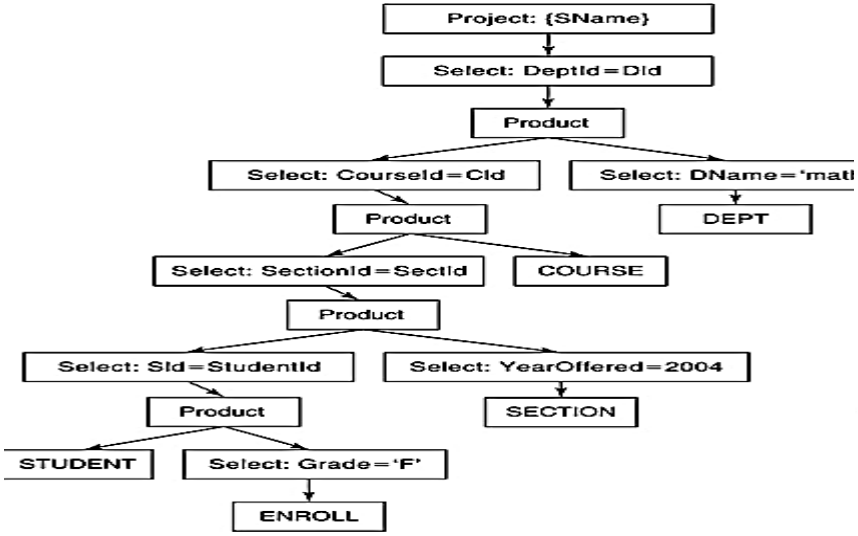


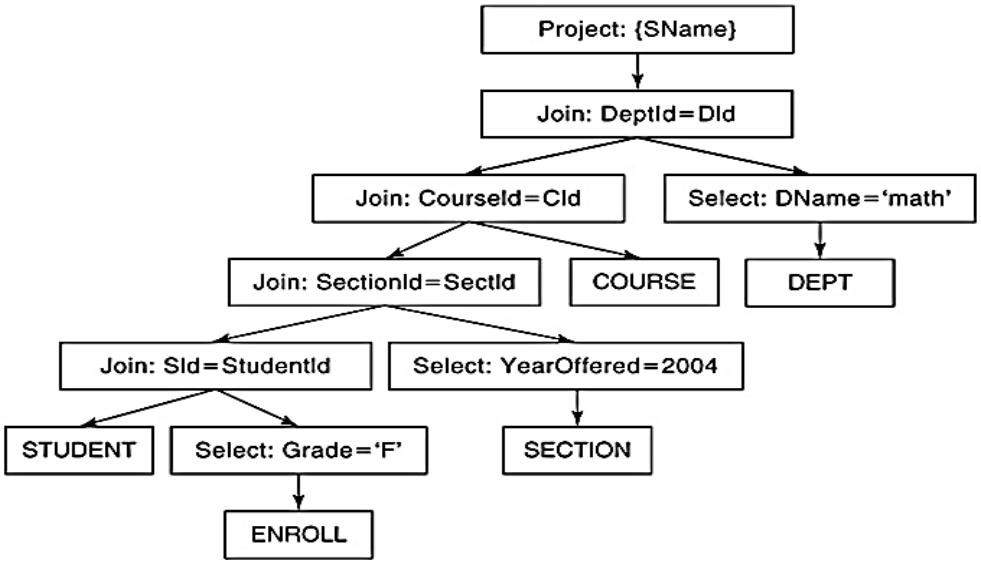




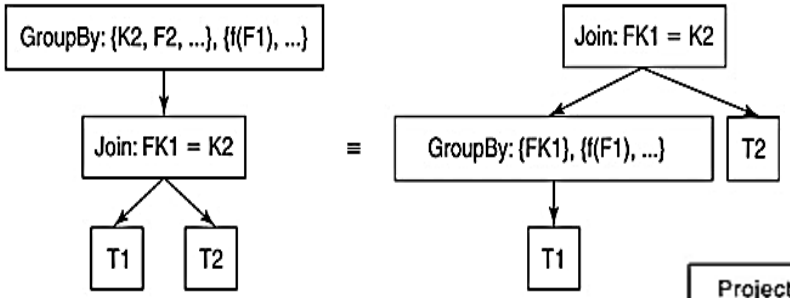
Ağaç Dönüşümleri-3: *join*

- Join(T1,T2,p) = SELECT(PRODUCT(T1,T2),p)

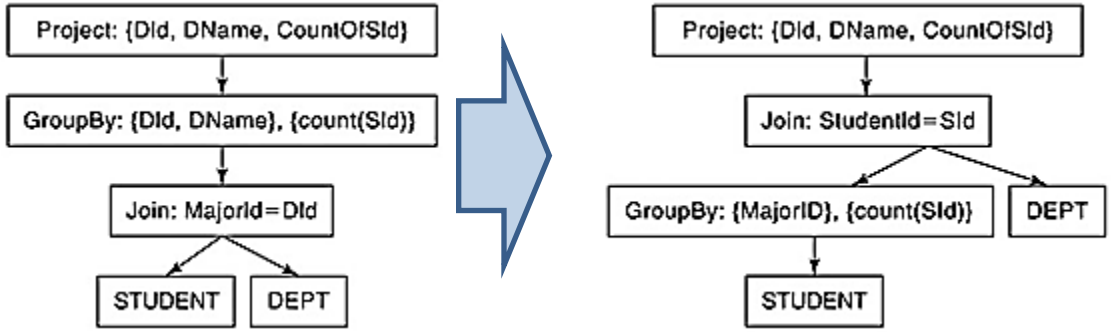




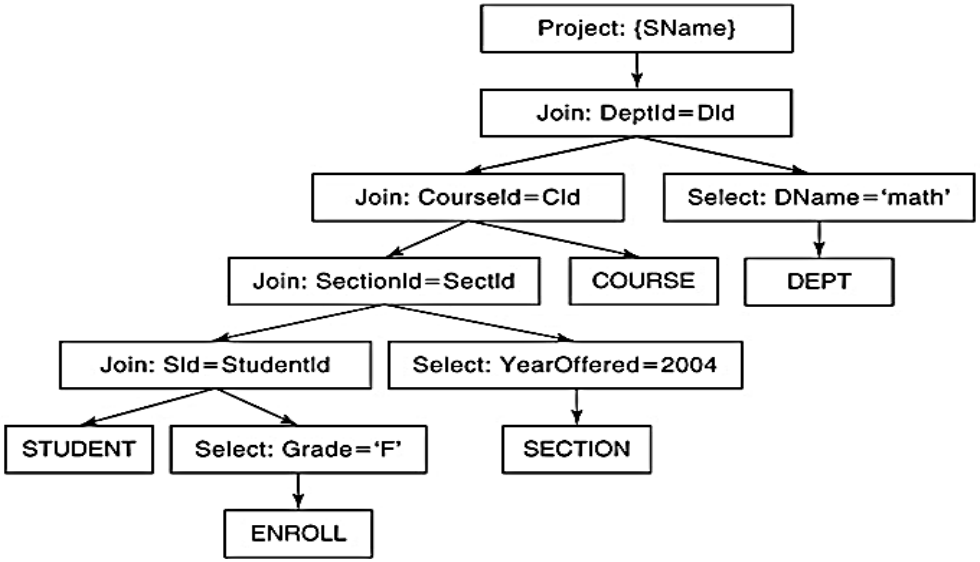
Ağac Dönüsümleri-4: *group by*

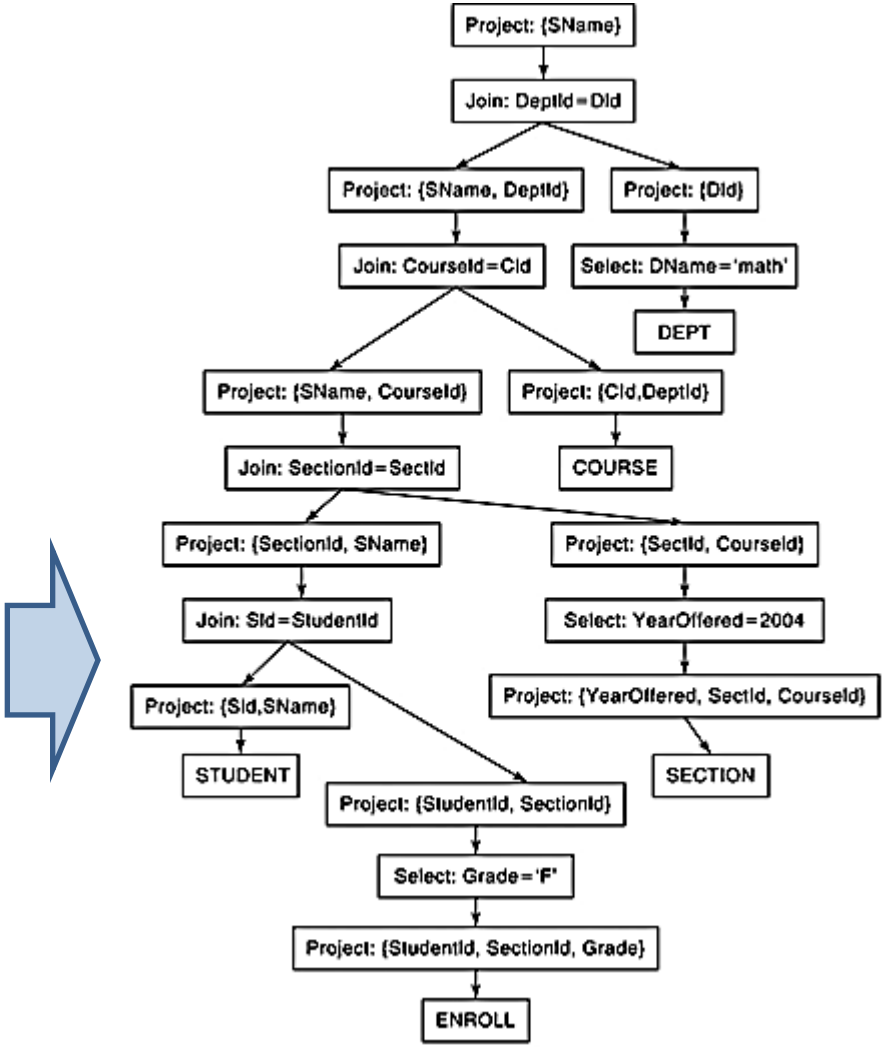


- Gruplama nitelikleri K2 anahtarını içermeli ve hepsi bir tablodan gelmeli.
- Kümeleme fonksiyonları sadece bir tablodan gelmeli



Ağaç Dönüşümleri-4: *project*

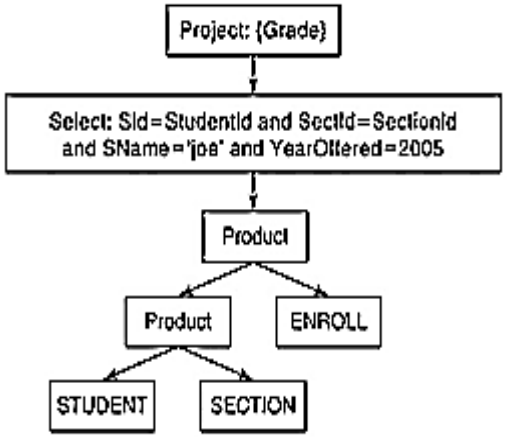


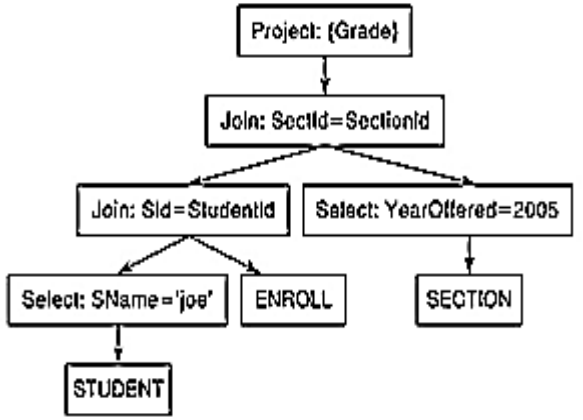


Ağaç kestirimi: *ağaç maliyet kestirimi*

- Blok erişim sayısı bilgisi olmadan ➔ ağac kestirimi
- 2 öngörü
  - Ağaç maliyeti belirleyen PRODUCT/JOIN sayısı
  - PRODUCT/JOIN maliyeti giriş kayıt sayıları toplamı ile **tahmin** edilebilir.

STUDENT	4,500	45,000	45,000 44,960 50 40	for F=SId for F=SName for F=GradYear for F=MajorId
SECTION	2,500	25,000	25,000 500 250 50	for F=SectId for F=CourseId for F=Prof for F=YearOffered
ENROLL	50,000	1,500,000	1,500,000 25,000 45,000 14	for F=EId for F=SectionId for F=StudentId for F=Grade





$(45000 + 25000) + (1125*10^6 + 1{,}5* 10^6)$

$=1.126.570.000$

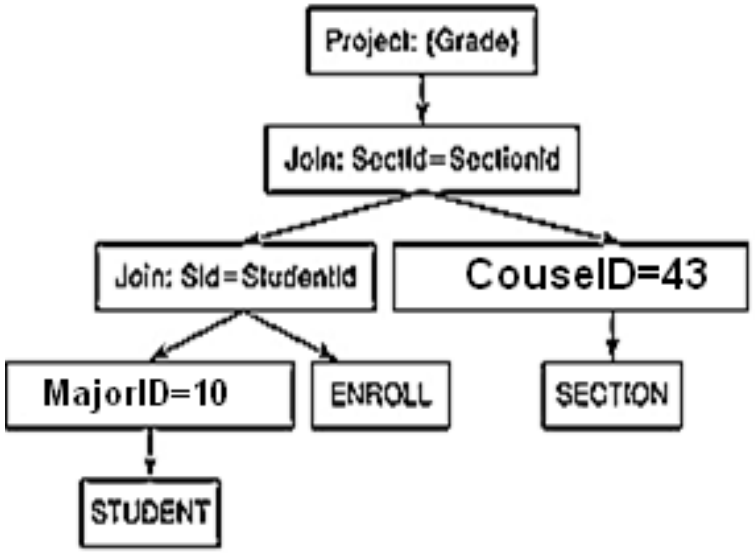
$(1+ 1{,}5* 10^6) + (34+500)$

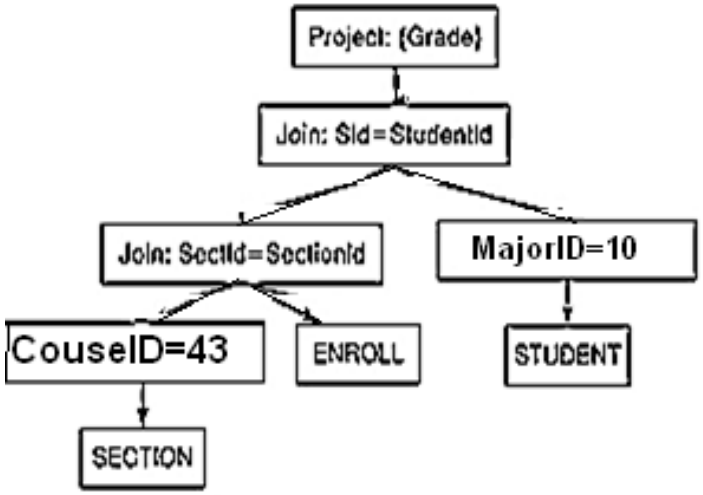
$=1.500.535$

8

Ağaç kestirimi: *ağaç maliyet kestirimi*

En düşük maliyetli ağaç → DUA → En düşük maliyetli PLAN





J1 maliyeti= 45000/40 + 1.5 milyon

J2 maliyeti= 1.5 milyon/40 + 25000/500

Toplam maliyet kestirimi= **1.538.675**

J1= ENROLL.StudentId idx, IndexJoin

B(J1) = 4500 + 37500

J2= MultiBuffer Product, k=5

B(J2) = 2500 +5+5 + 1*B(J1) = **44.510**

J1 maliyeti= 25000/500 + 1.5 milyon

J2 maliyeti= 1.5 milyon/500 + 45000/40

Toplam maliyet kestirimi= **1.504.175**

ENROLL.StudentId idx kullanamıyoruz!

J1= MultiBuffer Product, k=5

B(J1) = 2500+5+5+ 1*50.000

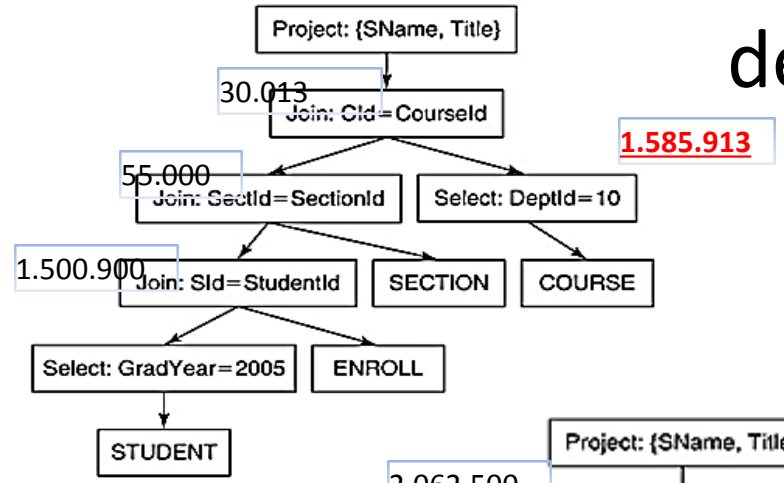
J2= MultiBuffer Product, k=113

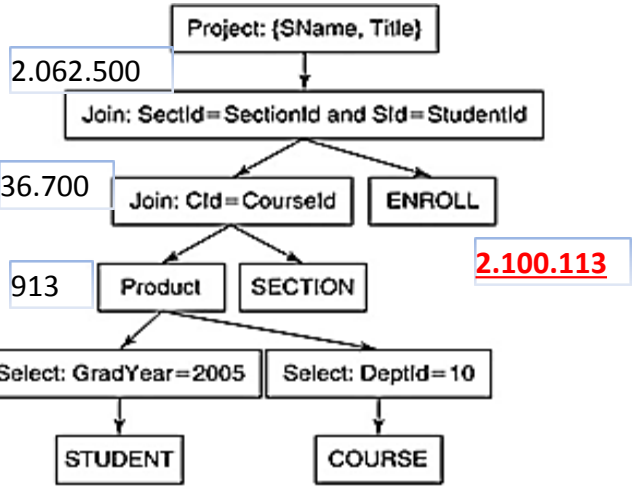
B(J2) = 4500 +113+113 + 1*B(J1) = **57.236**

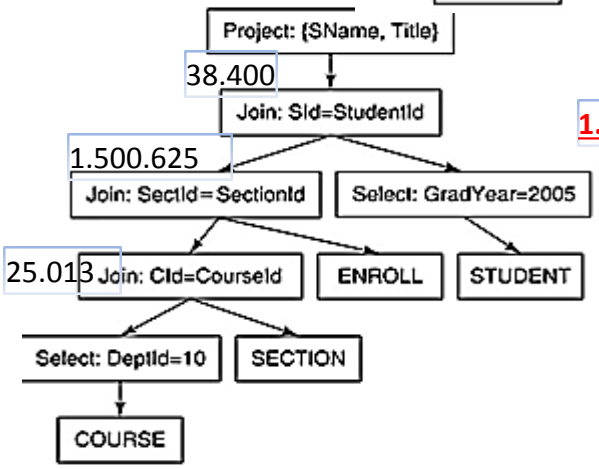
Ağaç kestirimi: Sezgiler

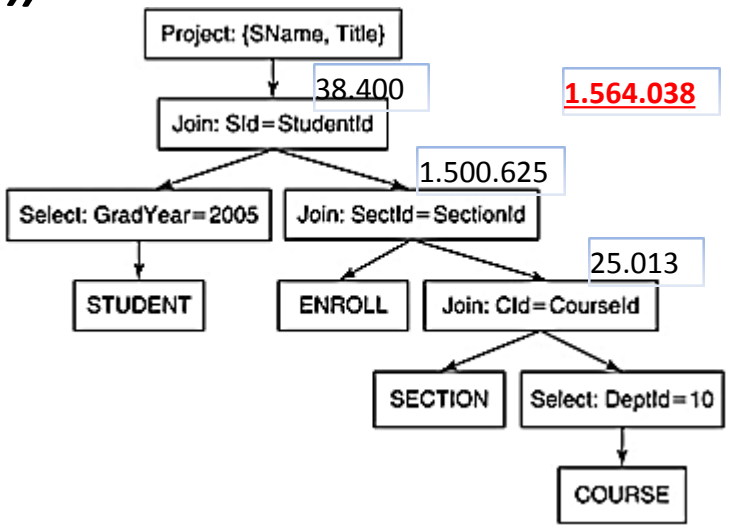
- Ağacın yapısını belirleyen SEZGİLER:
  - SEZGİ 1= «CNF formuna uygun yüklemdeki her bir alt-yüklem, ağaçta mümkün olan en aşağı pozisyona indirilir.»
  - SEZGİ 2= «Sorgudaki GROUP *BY(kümeleme fonksiyonları*), ağaçta mümkün olan en aşağı pozisyona indirilir.»
  - SEZGİ 3= «Ağaçtaki SELECT(PRODUCT(T1,T2),p) düğüm ikilileri yerine, JOIN(T1,T2,p) yazılır.»
  - SEZGİ 4= «Sadece SOLA-DAYALI olan ağaçları değerlendir.»
- **JOIN/PRODUCT sırasını (*N! farklı*) belirleyen SEZGİLER:**
  - **SEZGİ 5= «(*Mümkünse*) Sadece, «PRODUCT» içermeyen JOIN sıralarını değerlendir.»**
  - **SEZGİ 6-A=«En az sayıda kayıt üreten tabloyu önce seç.»**
  - **SEZGİ 6-B= «Filtreleme katsayısı (reduction factor) en fazla olan tabloyu önce seç.»**
  - Dinamik Programlama ile Kapsamlı sayım

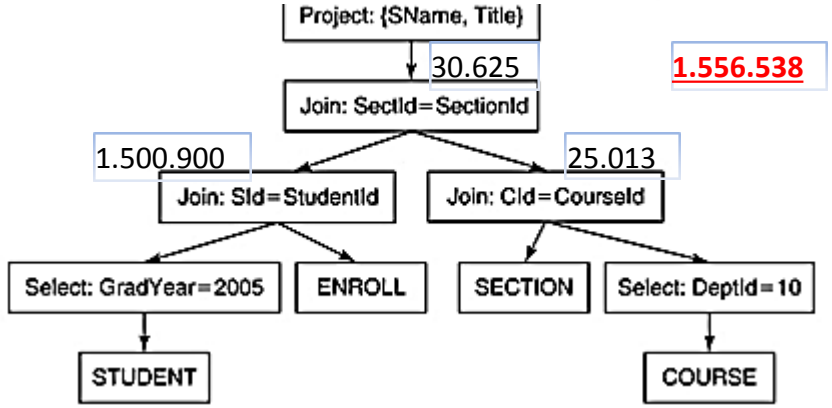
SEZGİ 4= «Sadece SOLA-DAYALI olan ağaçları değerlendir.»







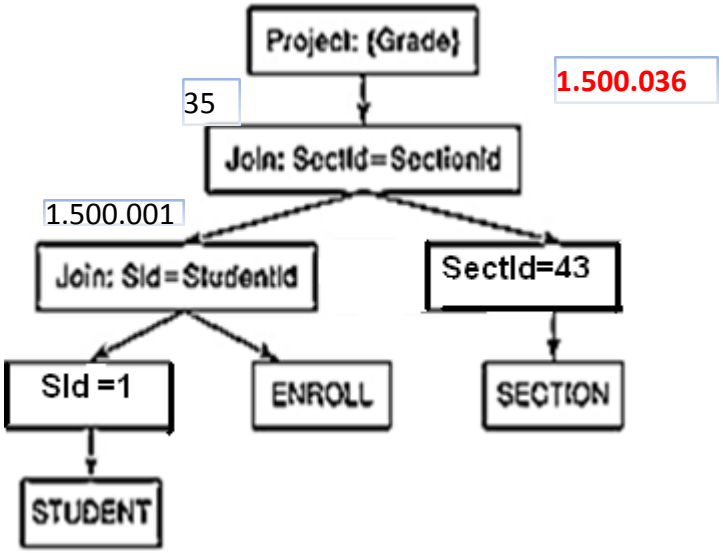


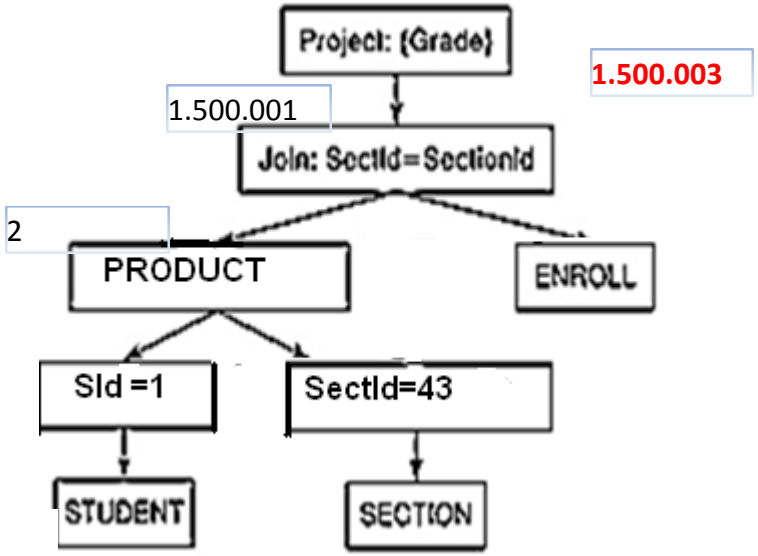


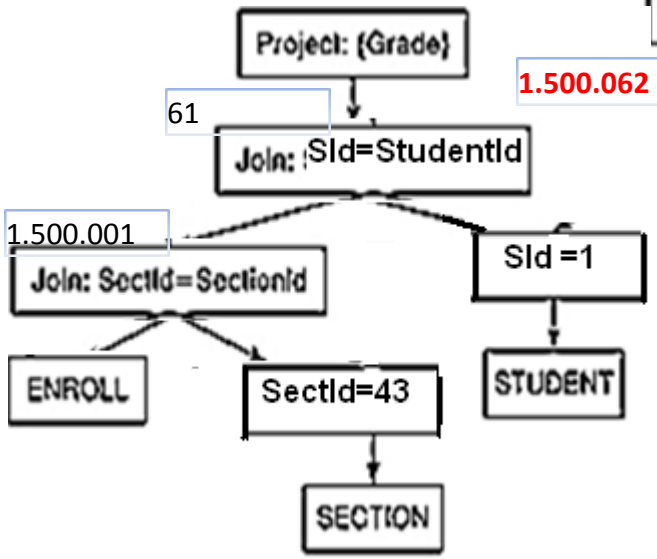
- SOLA-DAYALI AĞAÇLAR EN DÜŞÜK MALİYETE SAHİP OLMASA DA, EN İYİ PLANA SAHİP OLMA İHTİMALİ YÜKSEK.
- (2N)/N! ➔ N!

SEZGİ 5= «(*Mümkünse*) Sadece, «PRODUCT» içermeyen JOIN sıralarını değerlendir.»

- PRODUCT düğümü kaçınılmaz ise; mümkün oldukça *(ağaçta en yukarıya)* ertelenir.





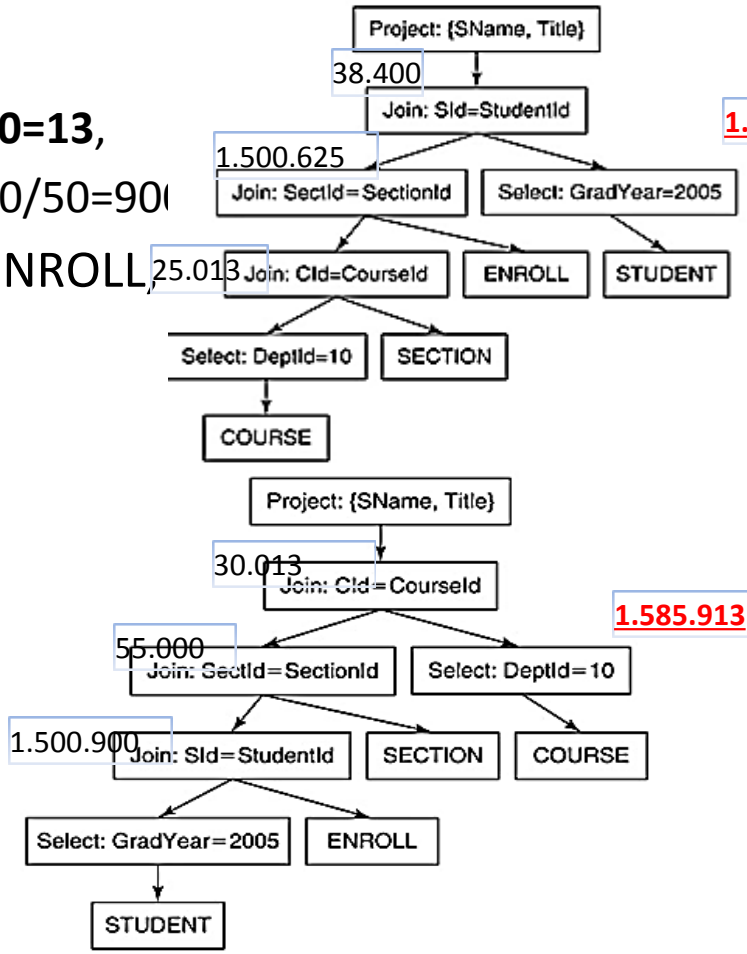


SEZGİ 6-A=«En az sayıda kayıt üreten tabloyu önce seç.»
SEZGİ 6-B= «Filtreleme katsayısı (reduction factor) en fazla olan tabloyu önce seç.»

```
select SName, Title
from STUDENT, ENROLL, SECTION, COURSE
where SId=StudentId and SectId=SectionId
and CId=CourseId and GradYear=2005 and DeptId=10
```

- SEZGİ-6A
  - **COURSE➔ 500/40=13**,
  - STUDENT➔ 45000/50=90(
- COURSE, SECTION,ENROLL, STUDENT

- SEZGİ-6B:
  - COURSE➔40,
  - **STUDENT➔50**
- STUDENT, ENROLL,SECTION, COURSE



STUDENT	4,500	45,000	45,000 44,960 50 40	for F=SId for F=SName for F=GradYear for F=MajorId
DEPT	2	40	40	for F=DId, DName
COURSE	25	500	500 40	for F=CId, Title for F=DeptId
SECTION	2,500	25,000	25,000 500 250 50	for F=SectId for F=CourseId for F=Prof for F=YearOffered
ENROLL	50,000	1,500,000	1,500,000 25,000 45,000 14	for F=EId for F=SectionId for F=StudentId for F=Grade

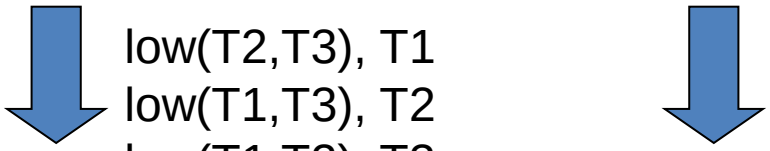
1.564.038

1.585.913

Dinamik Programlama ile Kapsamlı sayım

- S={T1,T2,…Tn}
- N=2 tablo,

lowest{T1,T2}, …..lowest{Ti,Tj}



- N=3 tablo,

lowest{[T1,T2,T3]},... lowest{[Ti,Tj,Tk}

- N=4 tablo,

lowest{[T1,T2,T3,T4}, ...lowest{[Ti,Tj,Tk,Tn}

……

- N=n-1 tablo

lowest{[T2,T3,….Tn]}, ….. lowest{[T1,T2,….Tn-1]}

- N=n tablo
  - En iyi JOIN sırası

Örnek: Student,Enroll,Section,Course

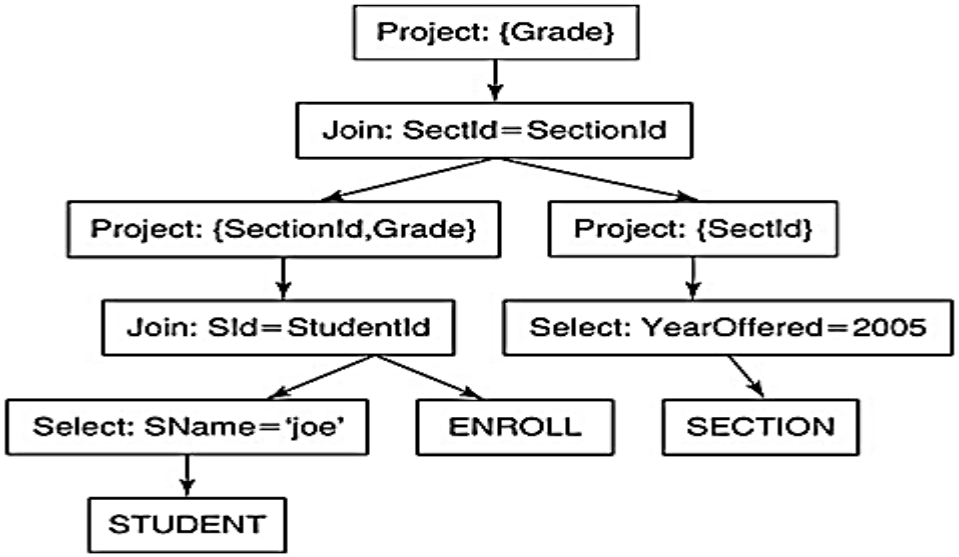
S	Partial Join Order	Cost	# Records
{ENROLL, STUDENT}	(STUDENT, ENROLL) (ENROLL, STUDENT)	1,500,900 1,500,900	30,000
{ENROLL, SECTION}	(SECTION, ENROLL) (ENROLL, SECTION)	1,525,000 1,525,000	1,500,000
{COURSE, SECTION}	(COURSE, SECTION) (SECTION, COURSE)	25,013 25,013	625
{SECTION, STUDENT}	(STUDENT, SECTION) (SECTION, STUDENT)	25,900 25,900	22,500,000
{COURSE, STUDENT}	(COURSE, STUDENT) (STUDENT, COURSE)	913 913	11,700
{COURSE, ENROLL}	(COURSE, ENROLL) (ENROLL, COURSE)	1,500,013 1,500,013	19,500,000

S	Partial Join Order	Cost	# Records
{ENROLL, SECTION, STUDENT}	(STUDENT, ENROLL, SECTION) (SECTION, ENROLL, STUDENT) (STUDENT, SECTION, ENROLL)	1,555,900 3,025,900 24,025,900	30,000
{COURSE, ENROLL, STUDENT}	(COURSE, STUDENT, ENROLL) (STUDENT, ENROLL, COURSE) (COURSE, ENROLL, STUDENT)	1,512,613 1,530,913 21,000,913	390,000
{COURSE, ENROLL, SECTION}	(COURSE, SECTION, ENROLL) (SECTION, ENROLL, COURSE) (COURSE, ENROLL, SECTION)	1,525,638 3,025,013 21,025,013	37,500
{COURSE, SECTION, STUDENT}	(COURSE, SECTION, STUDENT) (COURSE, STUDENT, SECTION) (STUDENT, SECTION, COURSE)	26,538 37,613 22,525,913	562,500

Join Order	Cost
(COURSE, SECTION, ENROLL, STUDENT)	1,564,038
(STUDENT, ENROLL, SECTION, COURSE)	1,585,913
(COURSE, STUDENT, ENROLL, SECTION)	1,927,613
(COURSE, SECTION, STUDENT, ENROLL)	2,089,038

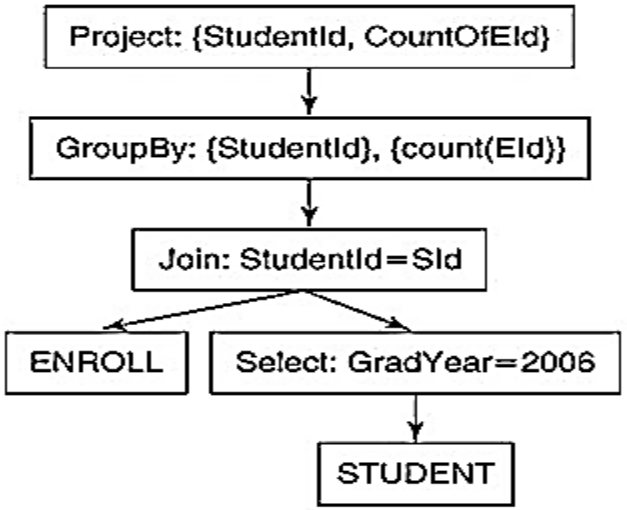
PLAN Kestirimi:SEZGİLER

- N düğümlü bir ağaç ➔ N*k farklı PLAN
- Öngörüler:
  - Aşağıdan-yukarı plan oluşumu
  - Blok erişim sayıları esas alınır.
  - Düğüm gerçeklemeleri birbirinden bağımsız.
- **SEZGİ 7=« SELECT düğümünü mümkünse INDEXSELECT ile gerçekleştir.»**
- **SEZGİ 8=«JOIN düğümünü mümkünse INDEXJOIN, değilse HASHJOIN veya MERGEJOIN ile gerçekleştir»**
- **SEZGİ 9=«Somutlaştırma düğümlerinin çocuğunu PROJECT düğümü yap»**



SimpleDB optimizasyonu

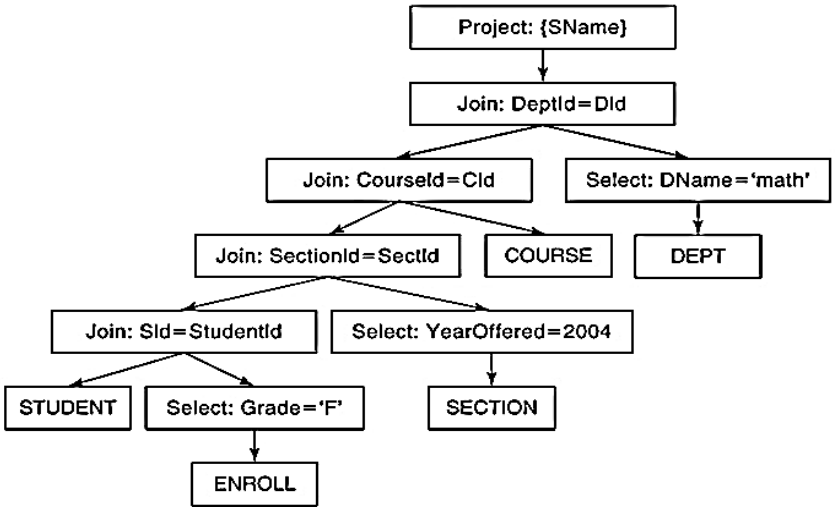
- İki aşama (Ağaç Kestirimi, PLAN kestirimi) eş zamanli.
  - Ağaç kestiriminde blok sayısı esas alınabilir.
  - Bir ağaç oluşturup saklamaya gerek olmaz.
  - Birbirinden bağımsız düğüm gerçeklemeleri



- SimpleDB.opt
  - HeuristicQueryPlanner (SEZGİ-6a)
  - TablePlanner

Örnek

```
select SName
from STUDENT, ENROLL, SECTION, COURSE, DEPT
where SId=StudentId and SectionId=SectId
and CourseId=CId and DeptId=DId
and DName='math' and Grade='F' and YearOffered=2004
```



- J1 kestirimi= 45000 + 1.5milyon/14
  - J1'den çıkan kayıt sayısı= 107.143
- J2 kestirimi= 107.143 + 25000/50
  - J2'den çıkan kayıt sayısı= 107.143/50 = 2143
- J3 kestirimi=2143 + 500
  - J3'den çıkan kayıt sayısı=2143
- J4 kestirimi =2143+1
  - J4'den çıkan kayıt sayısı=2143/40=54
- **TOPLAM maliyet kestirimi= 264.573**

- Ağacın maliyet kestirimi?
- Daha iyi Ağaç kestirimi?
  - SEZGİ-6A ile,
  - SEZGİ-6B ile,
  - (Sadece JOIN olanları içeren) Kapsamli Sayım ile.
    - Karşılık gelen PLAN Kestirimi
      - **SEZGİ-6A:**
        - **DEPT ➔ 1, COURSE ➔ 500**
        - **SECTION ➔ 500, ENROLL ➔ 107143**
        - **STUDENT➔ 45000**
        - **DEPT, COURSE, SECTION, ENROLL,STUDENT**
      - **SEZGİ-6B:**
        - **DEPT ➔ 40, COURSE ➔ 1**
        - **SECTION ➔ 50, ENROLL ➔ 14**
        - **STUDENT➔ 1**
        - **SECTION, ENROLL, COURSE, DEPT, STUDENT ,**

Örnek, «Stu,E,S,C,D» kapsamli sayim (*sadece JOIN*)

```
select SName
from STUDENT, ENROLL, SECTION, COURSE, DEPT
where SId=StudentId and SectionId=SectId
and CourseId=CId and DeptId=DId
and DName='math' and Grade='F' and YearOffered=2004
```

2'li	JOIN sırası	Maliyet Kestirimi	R(.)
Stu, E	**Stu,E** E,Stu	152.143	107.143
E,S	E,S **S,E**	107.643	2143
S,C	**S,C** C,S	1000	500
C,D	**D,C** C,D	501	13
Diğer bütün 2'liler (Stu,S; Stu,C; Stu,D; E,C; E,D; S,D) PRODUCT oluyor.			

3'li	JOIN sırası	Maliyet Kestirimi	R(.)
Stu, E,S	Stu,E,S **S,E,Stu**	259.786 154.786	2143
E,S,C	**C,S,E** S,E,C	108.643 110.286	2143
S,C,D	C,S,D **D,C,S**	1501 1014	13
Diğer bütün 3'lüler (Stu,S,C; Stu,C,D; Stu,E,C; Stu,E,D; Stu,S,D; E,S,D; E,C,D;) PRODUCT oluyor.			

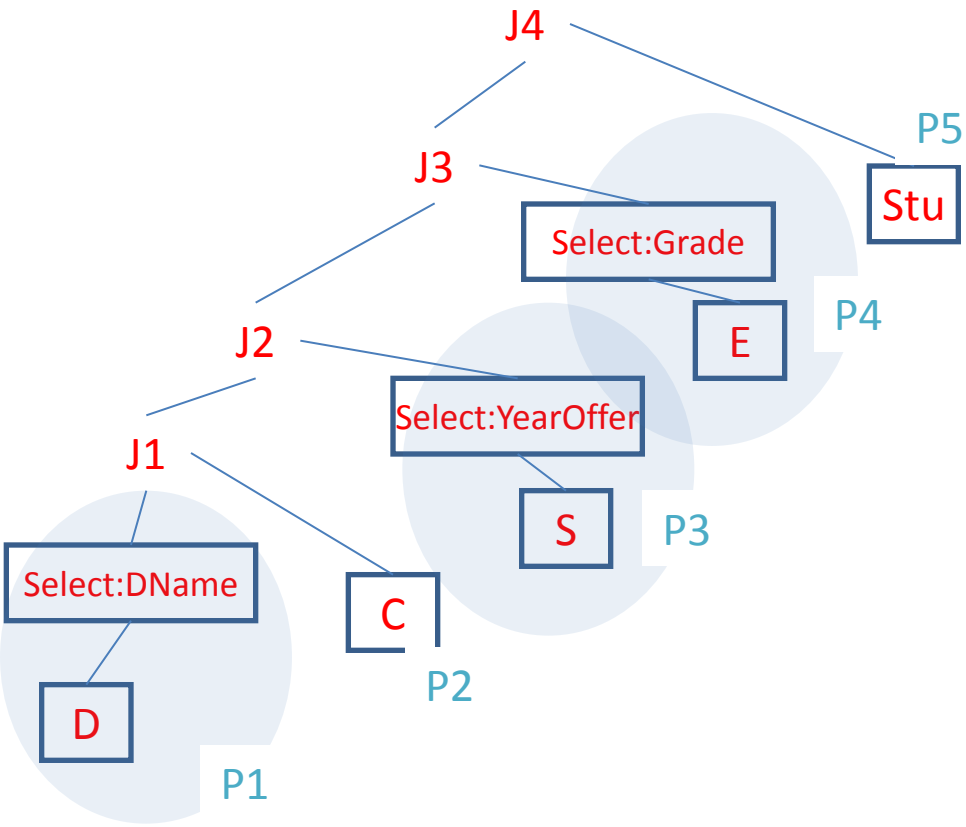
4'li	JOIN sırası	Maliyet Kestirimi	R(.)
Stu, E,S,C	S,E,Stu,C **C,S,E,Stu**	157.429 786	2143
E,S,C,D	C,S,E,D **D,C,S,E**	110.787 108.170	54
Diğer bütün 4'lüler (Stu,E,S,D; Stu,E,C,D; Stu,S,C,D) PRODUCT oluyor.			

5'li	JOIN sırası	Maliyet Kestirimi	R(.)
Stu, E,S,C,D	**D,C,S,E,Stu** C,S,E,Stu,D	**153.224** 157.930	54

Bulunan bu ağaç,
- dengeli ve sağa dayalı ağaçlar
- sola dayalı ağaçlardan PRODUCT içerenlerin

dışındaki bütün sola-dayalı ağaçlar arasında kestirilen en iyi ağaç.

Örnek: Kapsamlı Sayımda bulunan ağaç için(D,C,S,E,Stu) PLAN Kestirimi



Herhangi bir idx yoksa, HashJoin veya MergeJoin ile gerçekleniyor:

- B(lhs)=1, B(rhs)=25
  - J1: hashjoin(P2,P1, «DeptId», «Dıd»)
  - B(J1) = 2+1+1+25=29 *(k=3 tampon yeterli)*
  - R(J1) =13
- B(lhs)=1 *olabilir*, B(rhs)=2500/50
  - J2: hashjoin(P3, P_{J1}, «CourseID», «Cıd»);
  - B(J2) = B(J1)+ 1+1+2500= 2531 *(k=3 tampon yeterli)*
  - R(J2)=13
- B(lhs)=1 *olabilir*, B(rhs)=50.000/14
  - J3: hashjoin(P4, P_{J2}, «SectID», «SectionID»);
  - B(J3) = B(J2)+ 1+1+50.000= 52533 *(k=3)*
  - R(J2)=54
- B(lhs)=1 *olabilir*, B(rhs)=4.500
  - J4: hashjoin(P5, P_{J3}, «SID», «StudentID»);
  - B(J4) = B(J3)+ 1+1+4500= **57.035**

Örnek *(sola dayalı ağaçlar)*

- R,S,T tablolarının jon edilmesi 4! / 2! = 12 farklı ağaç ile olabilir. Bunların arasındabushy ağaç yok. 6 'sı sola dayalı, diğer 6'sı ise sağa dayalı olabilir.
- Sola dayalı ağaçlar:

(R,S),T

(S,R),T

(R,T),S

(T,R),S

(T,S),R

(S,T),R