

Veritabanı Yönetim Sistemleri

Dr. Öğr. Üyesi Ahmet Arif AYDIN

Query Optimization

Sorgu Optimizasyonu

- **Dosya Yapıları**
 - Pages, Disk space manager
 - Buffer manager
- **Index yapıları**
 - Hash & Tree based indexing
- **B+ Tree**
 - search, insert , delete

Sorgu Optimizasyonu (Query Optimization)

Veritabanında kayıtlı olan veriler üzerinde sorgulama işlemini gerçekleştirmek için SQL dili kullanılır

Sorgular (query)
SQL komutları (Select, insert, create, alter,)
ve
çeşitli operatörler
(and, or, like, - , + , between, exists, ...) 'den
oluşmaktadır.

Sorgu Optimizasyonu (Query Optimization)

SQL sorguları hesaplanırken
(işleme tabi tutulurken)
alternatif yollar ile
gerçekleştirilir.

Alternatif yollar
sorgu içerisinde kullanılan farklı operatörlerin
ve operatörlerin sağladığı işlemi gerçekleştiren
algoritmalarının farklı biçimde birleştirilmeleriyle
ortaya çıkmaktadır.

Sorgunun hesaplanması işleminde
en iyi (zaman, kaynak kullanımı) hesaplama planını
bulma sürecine sorgu optimizasyonu denir.

Sorgu Değerlendirme (Query Evaluation)

Sorgu İşleme alma - değerlendirme aşamaları (Query Evaluation)

1. Sorgu İlişkisel Cebir formuna dönüştürülür.
2. Kullanılan operatörlere göre işlem ağacı oluşturulur.
3. Her bir işlem için kullanılacak algoritmalar belirlenir.

Sistem Kataloğu (System Catalog)

- VTYS'lerinin sağladığı ortamda oluşturulan tablolar ve index yapısı **veri (data)** olarak tanımlanır.
- VTYS'de oluşturulan **her tablo ve index yapısı** bilgisini içeren **tanımlayıcı bilgilerden** oluşan **tablo koleksiyonu** bulunmaktadır.

Tablolar ve index yapısının kaydedilip yönetimin gerçekleştirildiği yapı **system catalog, catalog tables, data dictionary, catalog** olarak isimlendirilmektedir.

Sistem Kataloğu (System Catalog)

Catalogda bulunan bilgiler

- Tablolar (tablo ismi , kolon isimleri tipleri, index isimleri, PK, FK)
- indexler (index adı, yapısı (B+) , arama anahtarları)
- View (isim ve tanımlama)
- Cardinality (tablodaki tuple sayısı)
- Size (sayfa sayısı)
- Nkeys (index sayısı)
- Ağaç yapısının yüksekliği
- Index aralığı (range) min, max

Cataloglar periyodik olarak güncelleniyor (tabloların her işlemde değiştirilmesi ve güncellenmesi maliyetli)

Sistem Kataloğu (System Catalog)

SELECT * FROM pg_catalog.pg_tables;

	Data Output	Explain	Messages	Query History					
	schemaname name	tablename name	tableowner name	tablespace name	hasindexes boolean	hasrules boolean	hastriggers boolean	rowsecurity boolean	
1	pg_catalog	pg_statistic	ahmetaydin	[null]	true	false	false	false	
2	pg_catalog	pg_type	ahmetaydin	[null]	true	false	false	false	
3	public	ögrenci	ahmetaydin	[null]	true	false	true	false	
4	pg_catalog	pg_policy	ahmetaydin	[null]	true	false	false	false	
5	pg_catalog	pg_authid	ahmetaydin	pg_global	true	false	false	false	
6	public	kayıt	ahmetaydin	[null]	false	false	true	false	
7	public	dersler	ahmetaydin	[null]	true	false	true	false	
8	public	bölüm	ahmetaydin	[null]	true	false	true	false	
9	pg_catalog	pg_user_ma...	ahmetaydin	[null]	true	false	false	false	
10	pg_catalog	pg_subscript...	ahmetaydin	pg_global	true	false	false	false	
11	pg_catalog	pg_attribute	ahmetaydin	[null]	true	false	false	false	
12	pg_catalog	pg_proc	ahmetaydin	[null]	true	false	false	false	
13	pg_catalog	pg_class	ahmetaydin	[null]	true	false	false	false	
14	pg_catalog	pg_attrdef	ahmetaydin	[null]	true	false	false	false	
15	pg_catalog	pg_constraint	ahmetaydin	[null]	true	false	false	false	
16	pg_catalog	pg_inherits	ahmetaydin	[null]	true	false	false	false	
17	pg_catalog	pg_index	ahmetaydin	[null]	true	false	false	false	
18	pg_catalog	pg_operator	ahmetaydin	[null]	true	false	false	false	
19	pg_catalog	pg_opfamily	ahmetaydin	[null]	true	false	false	false	
20	pg_catalog	pg_opclass	ahmetaydin	[null]	true	false	false	false	

System kataloğu tablo koleksiyonudur

Sistem Kataloğu (System Catalog)

SELECT * FROM pg_catalog.pg_index;

	Data Output	Explain	Messages	Query History								
	indexrelid oid	indrelid oid	indnatts smallint	indisunique boolean	indisprimary boolean	indisexclusion boolean	indimmediate boolean	indisclustered boolean	indisvalid boolean	indcheckxmin boolean	indisready boolean	
1	2831	2830	2	true	true	false	true	false	true	false	true	
2	2833	2832	2	true	true	false	true	false	true	false	true	
3	2835	2834	2	true	true	false	true	false	true	false	true	
4	2837	2836	2	true	true	false	true	false	true	false	true	
5	2839	2838	2	true	true	false	true	false	true	false	true	
6	3599	3598	2	true	true	false	true	false	true	false	true	
7	2841	2840	2	true	true	false	true	false	true	false	true	
8	3440	3439	2	true	true	false	true	false	true	false	true	
9	2337	2336	2	true	true	false	true	false	true	false	true	
10	2847	2846	2	true	true	false	true	false	true	false	true	

Operatör Değerlendirmesi (Operator Evaluation)

- Her bir operator için birden fazla algoritma bulunmaktadır.
- Gerçekleştirilecek işlemde hangi algoritmanın kullanılacağı ve daha iyi sonuç vereceğini aşağıdaki faktörler belirler:
 - Tablolar da tutulan veri miktarı (size)
 - Varolan indexler ve sıralama (ordering)
 - Kullanılabilen bellek miktarı (RAM)

Operatör Değerlendirmesi (Operator Evaluation)

Sorgu işlemlerinde kullanılan algoritmalar aşağıdaki tekniklerle değerlendirme işlemlerini gerçekleştirirler

- **indexleme (indexing)**
 - WHERE ifadesinden sonra tanımlanan şartlar ile daha az veri üzerinde çalışılır (selections, joins)
- **Döngü (iteration)**
 - Tablolarda bulunan değerler veya sadece index tabloları taranır.
- **Bölümleme (partitioning)**
 - Sıralama (sorting) veya hash (hashing) ile üzerinde işlem yapılacak tuple sayısı değiştirilebilir.

Bağlantı Yolu (Access Path)

Tablolarda bulunan satırları (tuple) elde etme metoduna **access path** denir ve iki biçimde gerçekleşir.

1. File scan (tarama)
 2. Index + matching selection (index+ eşleştirme işlemi)
-
- Tree index: attribute ($<$, $>$, $=$, $,$) value tanımlamasına göre eşleştirme yapar.
 - (a,b,e) arama anahtarı içerisinde a ile başlayan sorgular gerçekleştirilir
 - (a) veya (a,b) aranır fakat (b,e) aranmaz..
 - Hash index: Arama indexinde attribute= value prensibine göre eşleştirme yapar.

Algorithms for Relational Operations

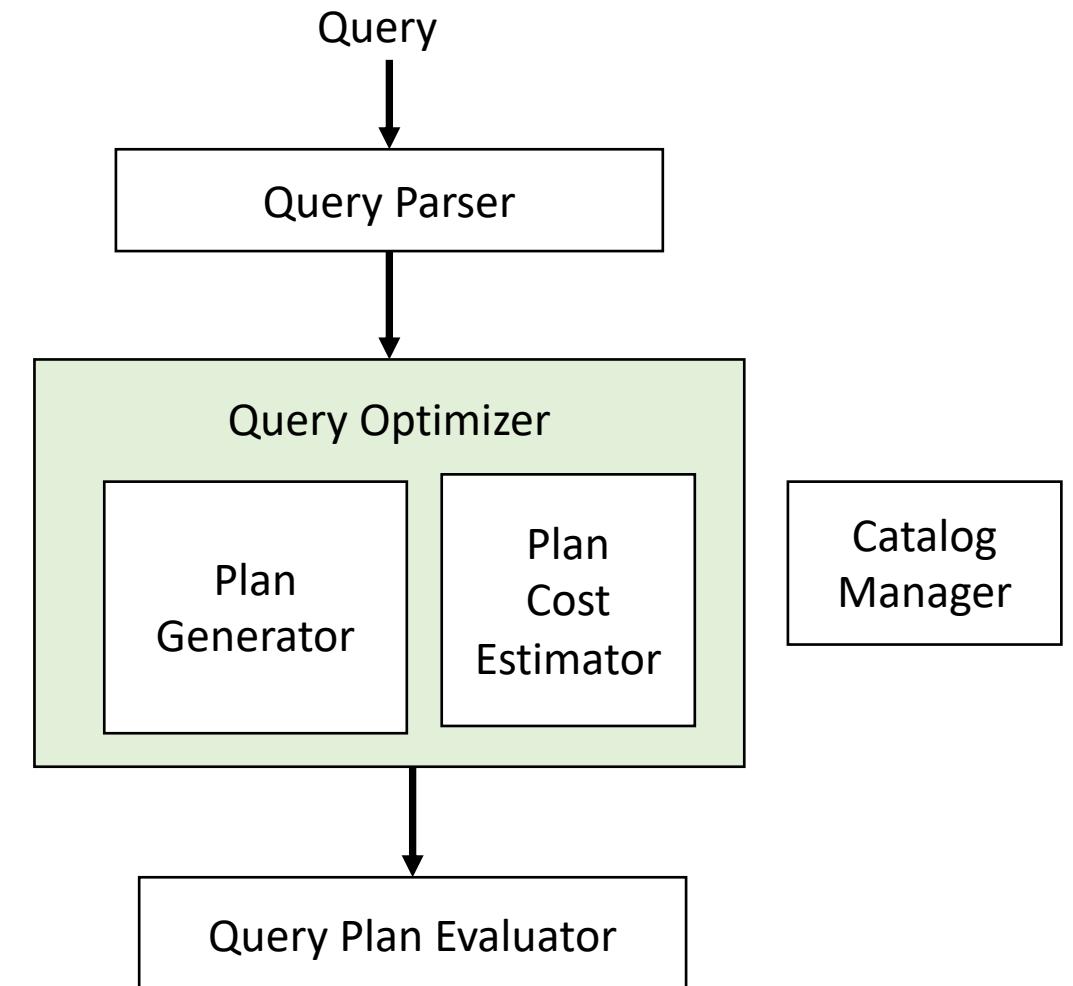
- **Seçim (Selection- σ)**
 - İstenilen alan index değilse tablo taranır.
- **İzdüşüm-Yansıtma (Projection- π)**: istenmeyen sütunları gizler
- **Join (\bowtie)**: maliyetli ve çok kullanılan bir işlemidir.

σ Tablo.colon (operatorler) değer (Tablo)

Query Optimizer

Query Optimizer sayesinde bir sorgunun
gerçekleştirme planları ve birinin maliyet
hesapları hesaplanır.

*a query is essentially treated as a σ
(selection), projection (π) - join (\bowtie)
algebra expression*

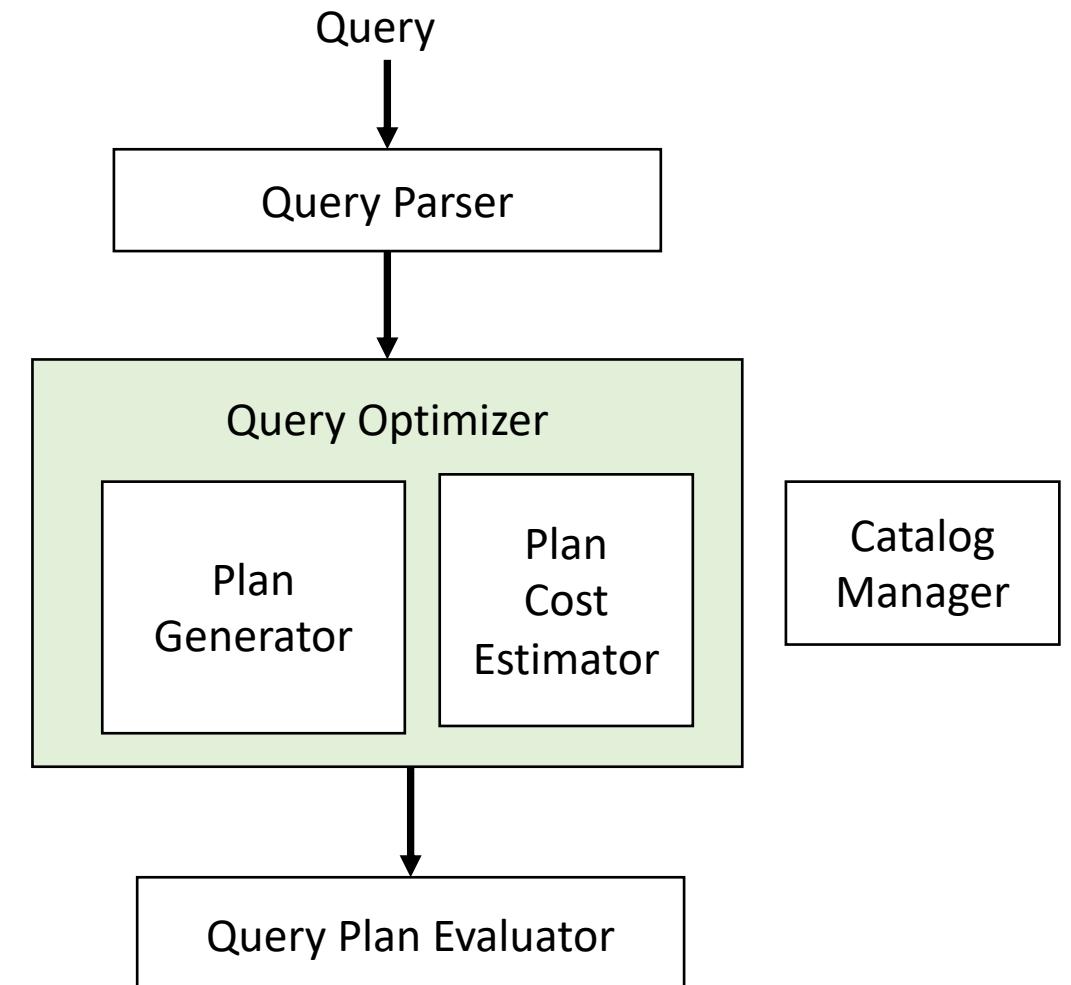


Query Optimizer

$A \bowtie B \bowtie C \bowtie D$

Kaç farklı biçimde join işlemi gerçekleştirilir?

?

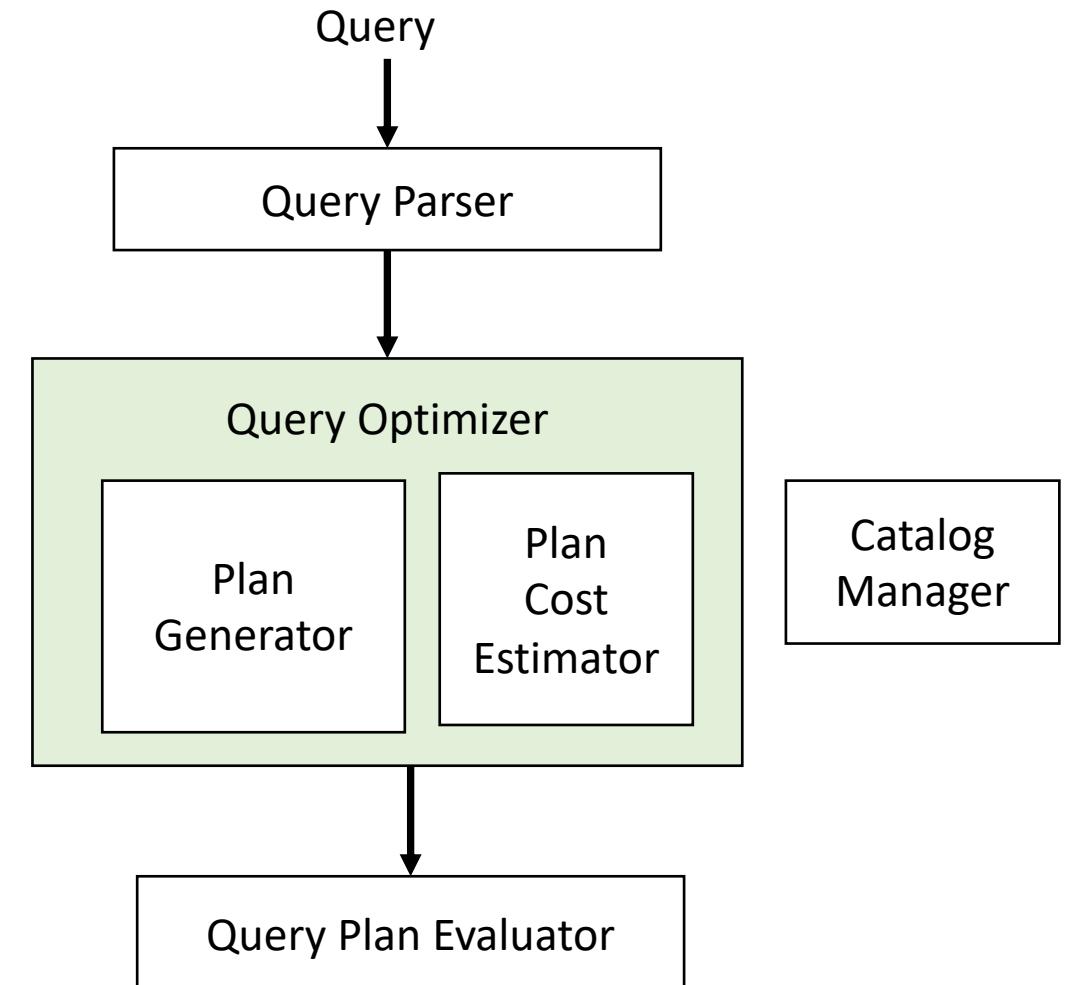


Query Optimizer

$A \bowtie B \bowtie C \bowtie D$

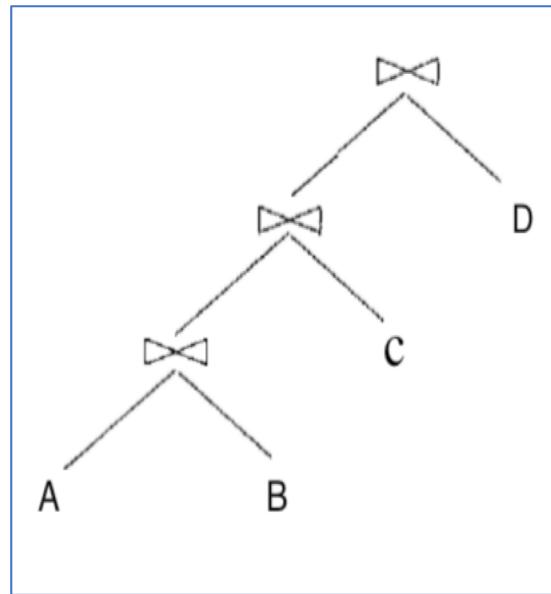
Kaç farklı biçimde join işlemi gerçekleştirilir?

$$4! = 24$$

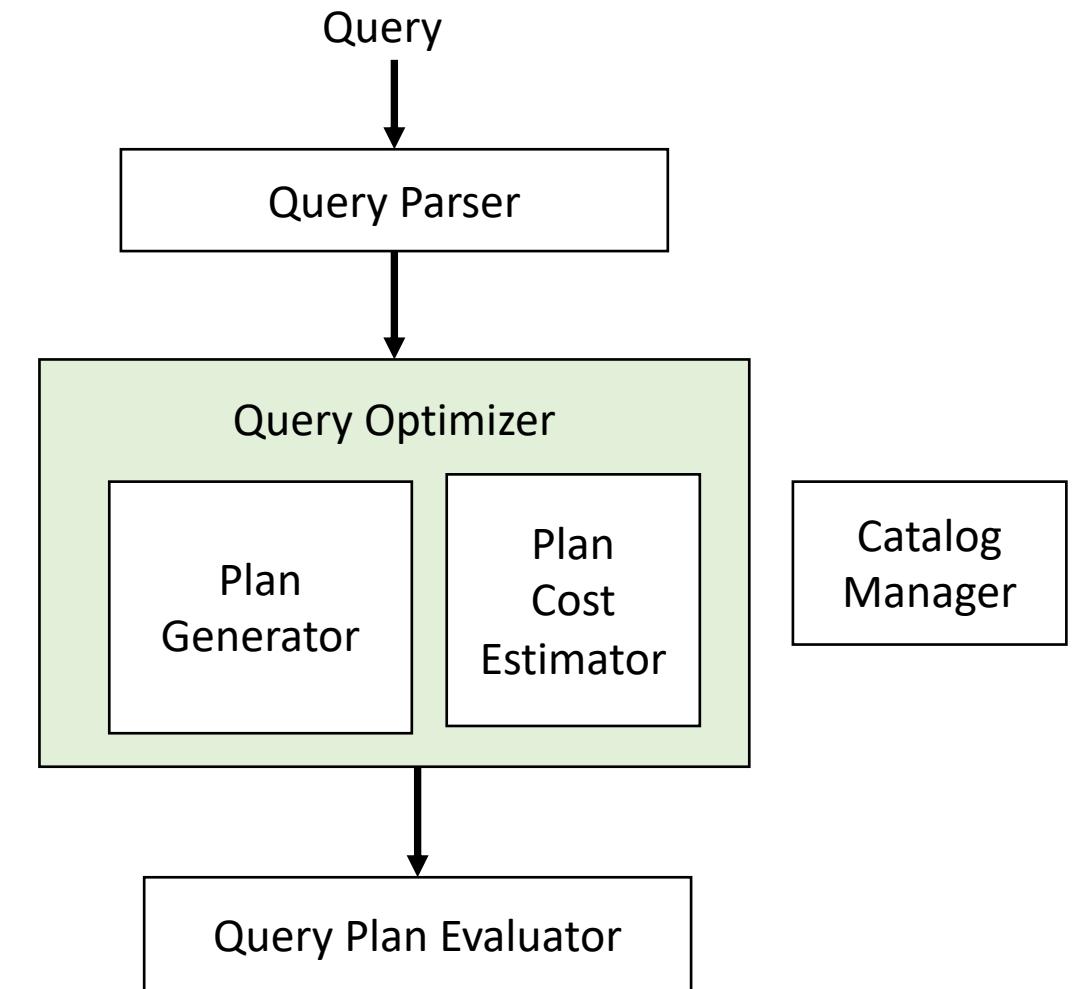


Query Optimizer

$A \bowtie B \bowtie C \bowtie D$



Önce A ile B birleştirilir, sonuc ile C, en son aşamada bir önceki sonuö ile D ile birleştirilir



Örnek tablo (ders kitabından)

Sailor

sid	sname	reyting	
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boat

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

s: sailor

b: boat

Query Evaluation

1

```
SELECT S.sname  
FROM   Reserves R, Sailors S  
WHERE  R.sid = S.sid  
       AND R.bid = 100 AND S.rating > 5
```

Query Evaluation

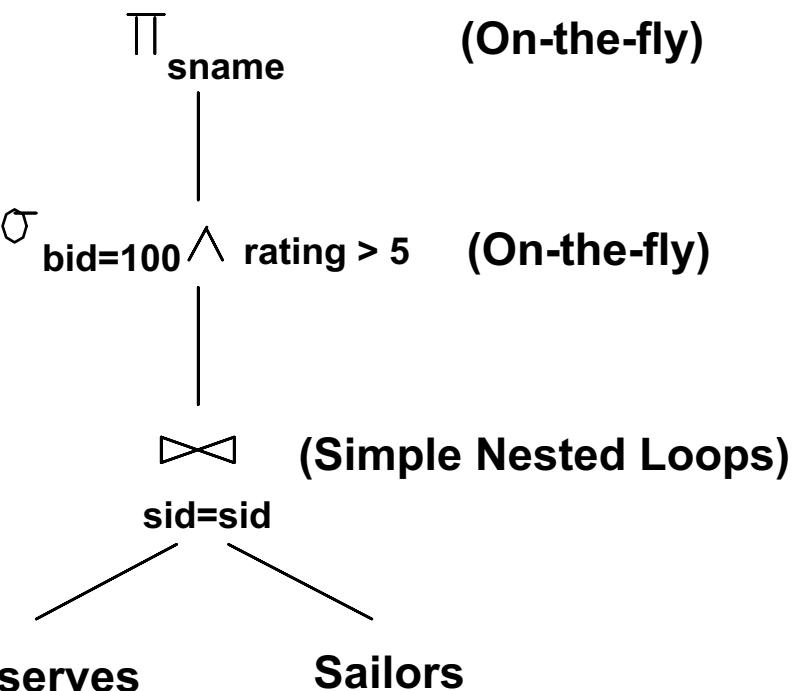
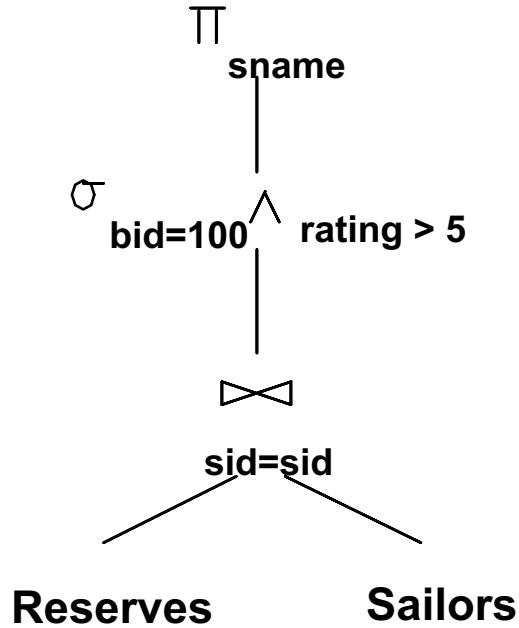
1
SELECT S.sname
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
AND R.bid = 100 AND S.rating > 5

2

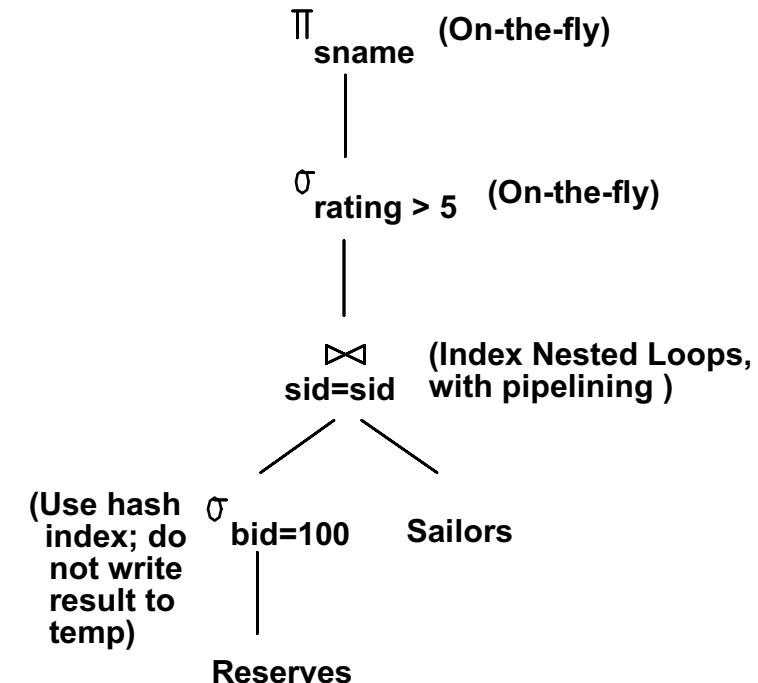
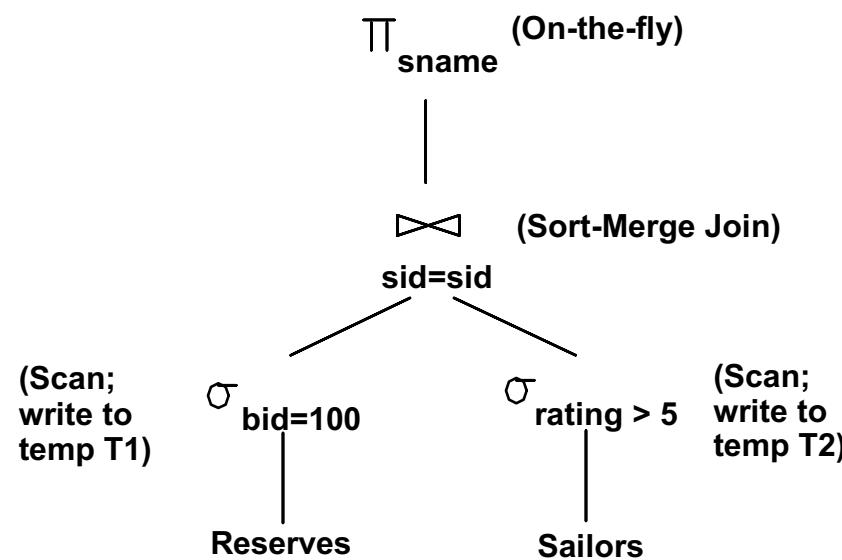
$\pi_{sname}(\sigma_{bid=100 \wedge rating>5}(Reserves \bowtie_{sid=sid} Sailors))$

Query Evaluation

3



Query Evaluation: alternative plans



PostgreSQL Sorgu Planı

```
vtys17=# explain analyze select öğrenci.ögrencino, öğrenci.ögrenciadı, öğrenci.yaş from öğrenci, kayıt where öğrenci.ögrencino = kayıt.ögrencino and kayıt.derskodu = ( Select derskodu from dersler where dersadı='Veritabanı Yönetim Sistemleri') order by yaş desc ;
```

QUERY PLAN

```
Sort  (cost=58.28..58.29 rows=4 width=40) (actual time=0.100..0.100 rows=2 loops=1)
  Sort Key: "öğrenci"."yaş" DESC
  Sort Method: quicksort  Memory: 25kB
  InitPlan 1 (returns $0)
    -> Seq Scan on dersler  (cost=0.00..18.12 rows=3 width=32) (actual time=0.010..0.011 rows=1 loops=1)
        Filter: ((dersadı)::text = 'Veritabanı Yönetim Sistemleri'::text)
        Rows Removed by Filter: 6
-> Hash Join  (cost=20.18..40.11 rows=4 width=40) (actual time=0.057..0.061 rows=2 loops=1)
  Hash Cond: ("öğrenci"."ögrencino" = "kayıt"."ögrencino")
    -> Seq Scan on "öğrenci"  (cost=0.00..17.20 rows=720 width=40) (actual time=0.013..0.014 rows=8 loops=1)
    -> Hash  (cost=20.12..20.12 rows=4 width=4) (actual time=0.033..0.033 rows=2 loops=1)
      Buckets: 1024  Batches: 1  Memory Usage: 9kB
        -> Seq Scan on "kayıt"  (cost=0.00..20.12 rows=4 width=4) (actual time=0.025..0.027 rows=2 loops=1)
            Filter: ((derskodu)::text = ($0)::text)
            Rows Removed by Filter: 10
```

Planning time: 0.880 ms

Execution time: 0.192 ms

(17 rows)

vtys17#

<http://www.vertabelo.com/blog/technical-articles/understanding-execution-plans-in-postgresql>