Chapter 12: 질의처리 (후반부-2)

개요

질의 비용산정

비용 산정을 위한 카타로그 정보

선택 연산

정렬

조인 연산

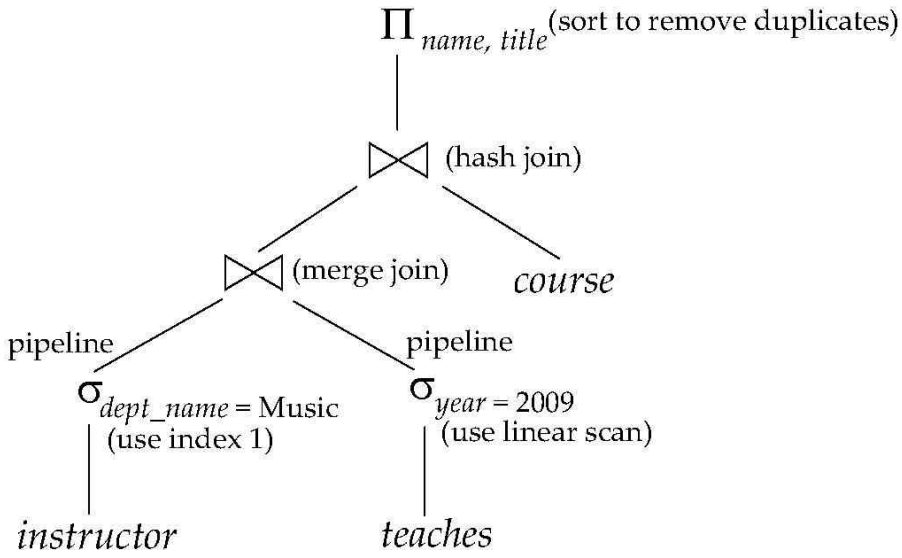
기타 연산

표현식의 평가

질의 처리 비용 산정

```
SELECT name, title
FROM instructor NATURAL JOIN teaches NATURAL JOIN course
WHERE instructor.dept_name='Music' and year=2009;
```

$\Pi_{name,\ title}(\sigma_{dept\_name=\text{"Music"} \wedge year = 2009}\ (instructor \bowtie (teaches \bowtie (course))))$



⋈

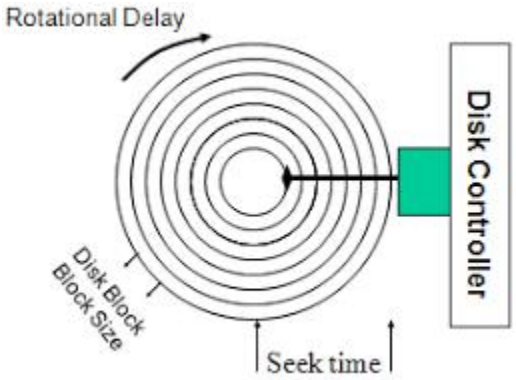
일반적으로 디스크로부터의 블록 전송 수(**number of block transfers**)를 실질적인 비용산정으로 사용된다.

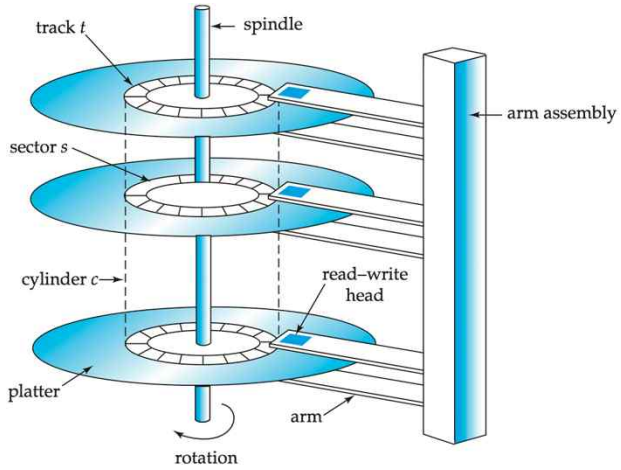
t_T – time to transfer one block

b개의 블록 전송: $b * t_T$

비용 산정을 위한 카타로그 정보

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000





- n_r : 릴레이션 r내의 튜플의 수
- b_r : r의 튜플들을 내포하고 있는 블록의 수
- s_r : r의 한 튜플의 바이트 단위의 크기
- f_r : r의 블록킹 요인 – 즉, 한 블록에 들어가는 r의 튜플 수

- **r의 튜플들이 파일내에 물리적으로** 함께 저장되면, 다음과 같다.

$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

- V(A, r) : 애트리뷰트 A에 대해 r에 나타나는 서로 다른 값의 수 ;

  $\Pi_A(r)$의 크기와 같다.

Example:

Block size 100byte로 가정

$n_{instructor} = 12$

$b_{instructor} = 12$

$s_{instructor} = 100$ byte

$f_{instructor} = 1$

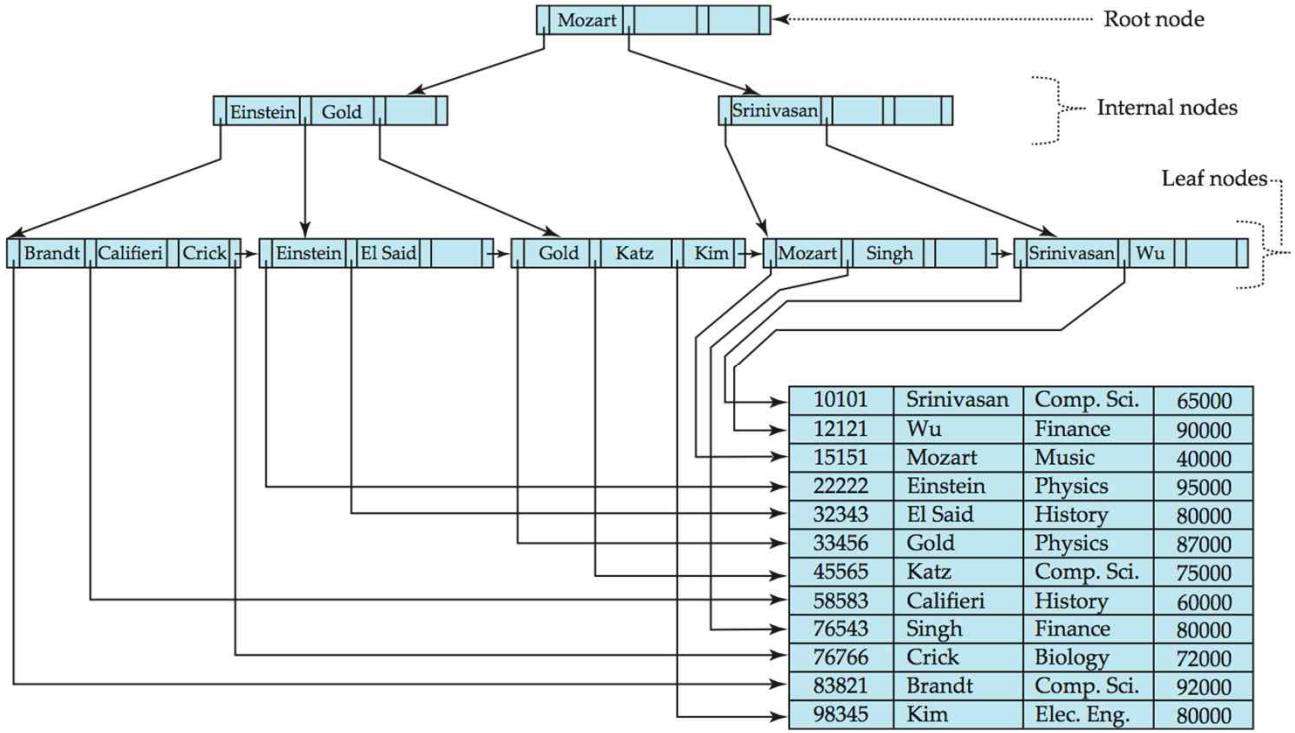
$b_{instructor}$ = ceiling(12/1) = 12

V(name, instructor) = 12

비용 산정을 위한 카타로그 정보 (계속)

- f_i : B^+ - 트리와 같은 트리 구조 인덱스에 있어, 인덱스 i의 내부 노드의 평균 전개
- h_i : 인덱스 i의 계층 수 – 즉, i의 높이이다.
  - (B^+ - 트리 같은) 인덱스에 있어, $h_i = \lceil \log_{f_i}(V(A, r)) \rceil$
  - 해쉬 인덱스에 있어 h_i는 1이다.

Example:
f_i=2
h_i=3



Example of B^+-Tree

죠인 연산 (Join Operation)

죠인을 실행하기 위한 알고리즘들은 다음과 같다

- 내포 – 루프 죠인 (Nested-loop join)
- 블록 내포 – 루프 죠인 (Block nested-loop join)
- 인덱스 내포 – 루프 죠인 (Indexed nested-loop join)
- 합병 – 죠인 (Merge-join)
- 해쉬 – 죠인 (Hash-join)

비용 산정에 따라 선택

예제

- 레코드 수 - instructor: 5,000 teaches: 10,000
- 블록 수 - instructor: 100 teaches: 400

죠인 연산 (Join Operation) 예제:

instructor

$n_{instructor}$=5000, $b_{instructor}$=100

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

⋈

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

teaches

$n_{teaches}$=10,000, $b_{teaches}$=400

Inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Pinance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Pinance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Pinance	90000	22222	PHY-101	1	Fall	2009
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2009
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2010
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2009
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2010
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2009
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2009
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2010
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2009
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2010
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2009
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

내포-루프 죠인 (Nested-Loop Join)

죠인 $r \bowtie s$ 을 계산하자

for each tuple t_r **in** r **do begin**
for each tuple t_s **in** s **do begin**
test pair (t_r, t_s) to see if they satisfy the join condition θ
if they do, add $t_r \bullet t_s$ to the result.
end
end

r을 죠인의 외부 릴레이션(**outer relation**)이라 하고 s를 내부 릴레이션(**inner relation**)이라 한다.

어떤 인덱스도 필요로 하지 않고 어떤 종류의 죠인 조건에도 사용될 수 있다.

두 릴레이션내 각 튜플의 쌍을 조사해야 하므로 비용이 많이 든다.

내포-루프 조인 (계속)

최악의 경우, 메모리에 각 릴레이션의 한 블록만을 수용할 수 있다면,

산정 비용은 블록 전송 수 : $n_r * b_s + b_r$
(이용 가능한 메모리가 최악의 경우라 가정했을 때)

Instructor 를 외부 릴레이션 으로 하면 :

- 5000 ∗ 400 + 100 = 2,000,100 block transfers,

teaches 를 외부 릴레이션으로 하면

- 10000 ∗ 100 + 400 = 1,000,400 block transfers

```
for each tuple t_r in r do begin
  for each tuple t_s in s do begin
        test pair (t_r,t_s) to see if they satisfy
         the join condition θ
        if they do, add t_r • t_s to the result.
 end
end
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

$n_{instructor}$=5000, $b_{instructor}$=100

10101 Shrinivasan …

12121 Wu …

10101 CS-101 …

10101 CS-315

메모리: 2블록 사이즈

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

$n_{teaches}$=10,000, $b_{teaches}$=400

내포-루프 조인 (계속)

보다 작은 릴레이션 전체가 메모리에 들어 간다면, 그것을 내부 레이션으로 사용한다.
그렇게 하면 비용 산정은 $b_s + b_r$ 의 블록 전송으로 줄어든다.

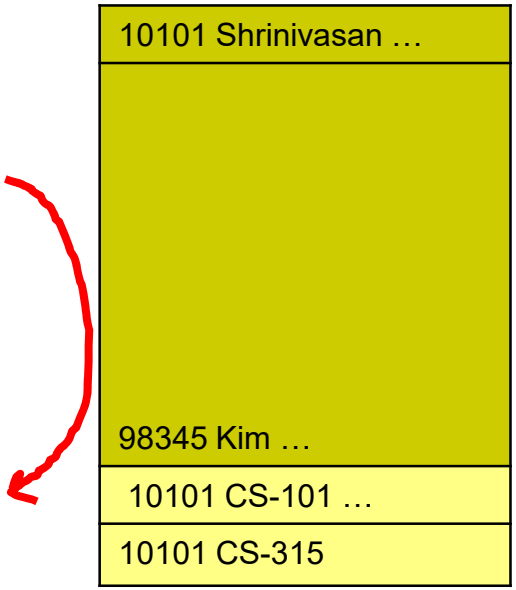
보다 작은 릴레이션(instructor) 전체가 메모리에 들어가면, 비용 산정은

100+400 = 500 블록 전송이 된다.

```
for each tuple t_r in r do begin
  for each tuple t_s in s do begin
        test pair (t_r,t_s) to see if they satisfy
         the join condition θ
        if they do, add t_r • t_s to the result.
 end
end
```

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

$n_{instructor}$=5000, $b_{instructor}$=100



메모리: 100 + 1 블록 사이즈

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

$n_{teaches}$=10,000, $b_{teaches}$=400

블록 내포 – 루프 죠인 (Block Nested-Loop Join)

Variant of nested-loop join in which every block of inner relation is paired with every block of outer relation.

for each block B_r **of** r **do begin**
  for each block B_s **of** s **do begin**
    for each tuple t_r **in** B_r **do begin**
      for each tuple t_s **in** B_s **do begin**
        Check if (t_r, t_s) satisfy the join condition
        if they do, add $t_r \bullet t_s$ to the result.
      end
    end
  end
end

블록 내포-루프 죠인 (계속)

최악의 경우: $b_r * b_s + b_r$ block transfers

내부 릴레이션 s내 각 블록은 외부 릴레이션 각 블록마다 한번씩만 읽힌다.

최상의 경우: $b_r + b_s$ block transfers.

```
for each block B_r of r do begin
    for each block B_s of s do begin
        for each tuple t_r in B_r do begin
            for each tuple t_s in B_s do begin
                Check if (t_r, t_s) satisfy the join condition
                if they do, add t_r • t_s to the result.
            end
        end
    end
end
```

최악의 경우: 100*400 + 100 = 40100
최상의 경우: 100+400 = 500

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

$n_{instructor}$=5000, $b_{instructor}$=100

10101 Shrinivasan …

12121 Wu …

10101 CS-101 …

10101 CS-315

메모리: 2블록 사이즈

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

$n_{teaches}$=10,000, $b_{teaches}$=400

인덱스 내포 – 루프 조인 (Indexed Nested-Loop Join)

내부 릴레이션의 조인 애트리뷰트에 인덱스가 있고 조인이 동등 – 조인 또는 자연 조인이면, 보다 효율적인 인덱스 탐색이 파일 스캔을 대치할 수 있다.

조인을 계산하기 위해서 독자적으로 인덱스를 구축할 수도 있다.

외부 릴레이션 r내의 각 튜플 t_r에 대해, 인덱스를 사용해 튜플 t_r과 조인 조건을 만족하는 s내의 튜플을 탐색한다.

최악의 경우 : 버퍼에 r의 한 페이지와 인덱스의 한 페이지만을 위한 공간을 가진다.

조인 비용(블록 전송 비용만 고려하는 경우): $b_r + n_r * c$

c는 조인 조건을 사용하고 있는 s상의 단일 선택 비용이다

r과 s 모두에 인덱스가 있다면, 보다 적은 튜플을 가진 것을 외부 릴레이션으로 사용한다.

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

$n_{teaches}$=10,000, $b_{teaches}$=400

10101 CS-101 …

10101 CS-315 …

10101 xxxxx 12121 xxxx

15151 xxxxx 22222 xxxxx

10101	10101	Srinivasan	Comp. Sci.	65000
12121	12121	Wu	Finance	90000
15151	15151	Mozart	Music	40000
22222	22222	Einstein	Physics	95000
32343	32343	El Said	History	60000
33456	33456	Gold	Physics	87000
45565	45565	Katz	Comp. Sci.	75000
58583	58583	Califieri	History	62000
76543	76543	Singh	Finance	80000
76766	76766	Crick	Biology	72000
83821	83821	Brandt	Comp. Sci.	92000
98345	98345	Kim	Elec. Eng.	80000

$n_{instructor}$=5000, $b_{instructor}$=100

최악의 경우: 400 + 10000*2 = 20,400

인덱스 내포 – 루프 죠인 예제

instructor 를 외부 릴레이션으로 instructor⋈*teaches* 를 계산한다.

$n_{instructor}$=5000, $b_{instructor}$=100 $n_{teaches}$=10,000, $b_{teaches}$=400

teaches 가 죠인 애트리뷰트 *ID* 상에 primary B^+ - 트리 인덱스를 가지고, 각 인덱스 노드에는 20개의 엔트리가 포함된다고 가정한다.

teaches 에는 10,000개의 튜플이 있으므로, 트리의 높이는 4 ($\log_{10}10000$= 4) 이고 실제 데이터를 찾으려면 한 번의 액세스가 더 필요하다.

Instructor 는 5,000 개의 튜플을 가진다.

block nested loops join의 비용: $b_r * b_s + b_r$

100*400 + 100 = 40,100 의 블록 전송

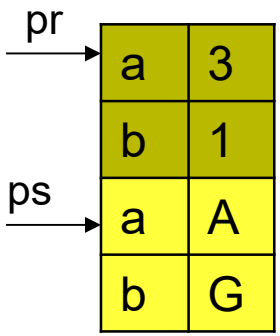
- 최악의 경우의 메모리를 가정
- 메모리가 증가하는 경우, 비용이 현저히 낮아질 수 있음

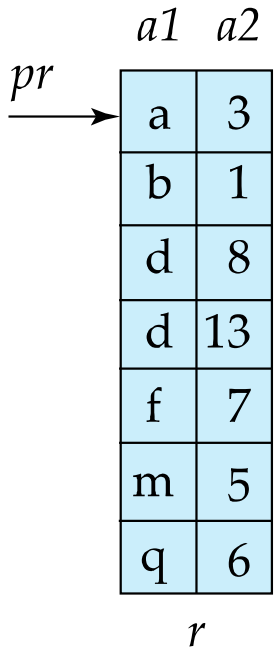
indexed nested loops join의 비용: $b_r + n_r * c$

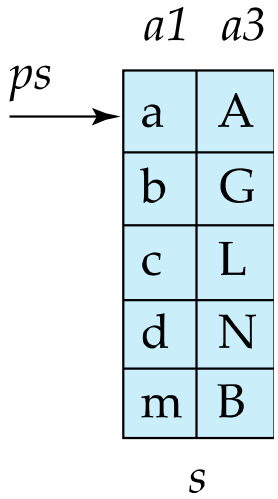
100 + 5000 * 5 = 25,100의 블록 전송

합병 죠인 (Merge-Join)

1. 먼저 양 릴레이션을 그들의 죠인 애트리뷰트에 따라 정렬한다 (정렬되어 있지 않으면).
2. 정렬된 릴레이션을 합병한다.
   - 죠인 절차는 정렬 – 합병 알고리즘의 합병 단계와 유사하다.
   - 주된 차이점은 죠인 애트리뷰트 내의 중복 값을 처리하는 것이다. 즉, 죠인 애트리뷰트상에 같은 값을 가진 각 쌍이 매치되도록 하여야 한다. (detailed algorithm in book)







합병 죠인 (계속)

동등 – 죠인과 자연 죠인에만 사용될 수 있다.

각 튜플이 한번씩만 읽힐 필요가 있으면, 그 결과 각 블록 또한 한 번만 읽힌다.

따라서, 비용은 (탐색 비용을 무시하는 경우):

- $b_r + b_s$ 블록 전송 비용
- sorting 비용 (릴레이션이 정렬되어 있는 않은 경우).

해쉬 죠인 (Hash-Join)

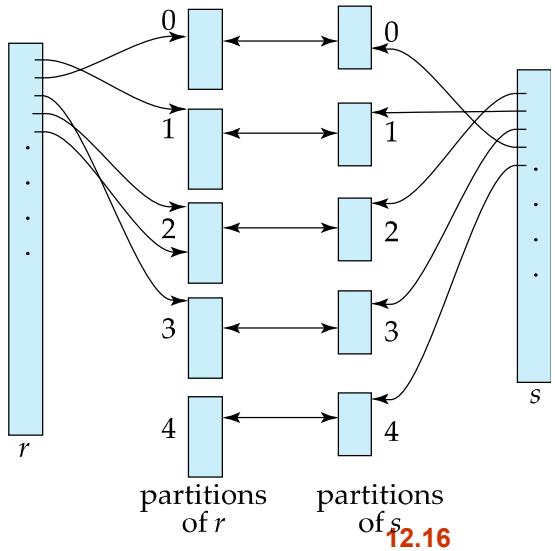
동등 – 죠인과 자연 죠인에 적용 가능하다.

양쪽 릴레이션의 튜플들을 죠인 애트리뷰트에 같은 해쉬 값을 가지는 집합으로 분할하기 위해 해쉬 함수 h를 사용한다.

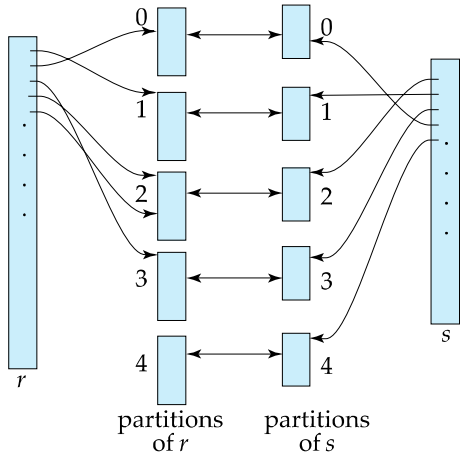
h는 JoinAttrs 값들을 {0, 1, …, n}로 대응시킨다

(JoinAttrs는 자연 죠인에 사용되는 r과 s의 공통 애트리뷰트를 나타낸다).

- $r_0, r_1, \ldots, r_n$ 은 r 튜플들의 분할을 나타내며, 각각이 처음에는 비어 있다.
  - $t_r \in r$인 각 튜플은 분할 r_i 에 들어가며, $i = h(t_r[\text{JoinAttrs}])$이다.
- $s_0, s_1, \ldots, s_n$ 은 s 튜플들의 분할을 나타내며, 각각이 처음에는 비어 있다.
  - $t_s \in s$인 각 튜플은 분할 s_i 에 들어가며, $i = h(t_s[\text{JoinAttrs}])$이다.



해쉬 조인 (계속)



H (10101) = (1+0+1+0+1)mod3 = 0
H(12121) = (1+2+1+2+1)mod3 = 1
H(15151) = 1
H(22222) = 1
H(32343)= 0

…
H(98345) = (9+8+3+4+5)mod3 = 2

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

10101, Shirinivasan
32343, El Sad, …
…

12121, Wu, …
15151, Mozart, ..
…

76766, Crick, …
98345, Kim, …

10101, CS-101, …
10101, CS-315, …
…

12121, Fin…
15151, MU…
…

76766, BIO-101
76766, BIO-301
98345, EE-181, …

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

해쉬 죠인 (계속)

r_i 내의 r 튜플들은 s_i 내의 s 튜플들과만 비교될 필요가 있다 ; 다른 분할 내의 s 튜플과는 다음과 같은 이유로 비교될 필요가 없다 :

- 죠인 조건을 만족하는 하나의 r 튜플과 하나의 s 튜플은 죠인 애트리뷰트에 대해 같은 값을 갖는다.
- 그 값이 어떤 값 i로 해쉬되면, r 튜플은 r_i 내에 있어야 하고 s 튜플은 s_i 내에 있어야 한다.

해쉬-죠인 알고리즘

- r과 s의 해쉬 죠인은 다음과 같이 계산된다.
  1. 해쉬 함수 h를 사용해 릴레이션 s를 분할한다. 릴레이션을 분할할 때, 메모리의 한 블록은 각 분할의 출력 버퍼로 남겨둔다.
  2. 위와 유사하게 r을 분할한다.
  3. 각 i에 대해 :
     (a) s_i를 메모리에 넣고 죠인 애트리뷰트를 사용해 메모리내에 해쉬 인덱스를 구축한다. 이 해쉬 인덱스는 앞의 h와는 다른 해쉬 함수를 사용한다.
     (b) 디스크에서 하나씩 r_i 내의 튜플들을 읽는다. 각 튜플 t_r에 대하여 메모리내 해쉬 인덱스를 사용해 s_i 내의 매칭하는 각 튜플 t_s를 찾는다.

- 릴레이션 s를 구축 입력(**build input**)이라 하고 r을 탐색 입력(**probe input**)이라 부른다.
- 반복 분할이 요구되지 않으면 비용은 약 $3(b_r + b_s)$이 된다.

해쉬 - 죠인 비용 산정의 예제

instructor ⋈ *teaches*

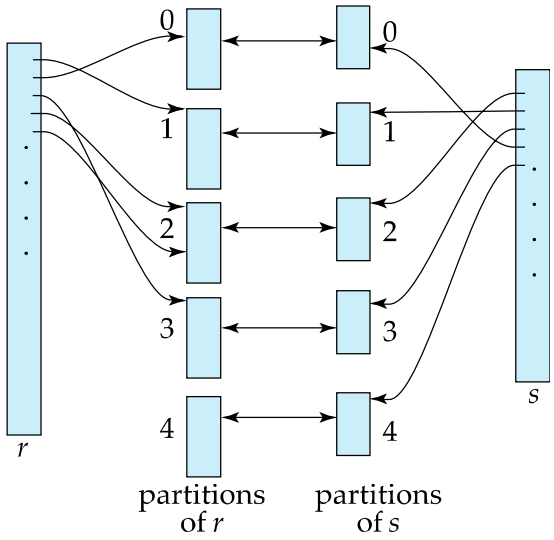
메모리 크기 *M*은 20 블록분으로 가정한다.

$b_{instructor}$= 100 $b_{teaches}$ = 400.

Instructor 가 구축 입력으로 사용될 것이다. 그것은 각각의 크기가 20블록짜리 다섯 개의 분할로 분할된다. 이 분할은 한 패스로 이루어질 수 있다.

이와 유사하게, *teaches* 는 80블록짜리 5개의 분할로 분할되며, 이 또한 한 패스로 이루어질 수 있다.

그러므로, 총 비용은 $3(b_r + b_s)$ = 3(100 + 400) = 1500 블록 전송이 된다



복합 조인

다음과 같은 논리곱 조건을 가진 조인 :

$$r \bowtie_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n} s$$

nested loops/block nested loops 조인 방식을 사용하거나,

단순한 조인들 $r \bowtie_{\theta i} s$ 중 하나의 조인 결과를 계산하고,

- 마지막 결과에는 다음과 같은 나머지 조건을 만족하는 중간 결과 내의 튜플들을 포함시킨다.

$$\theta_1 \wedge \ldots \wedge \theta_{i-1} \wedge \theta_{i+1} \wedge \ldots \wedge \theta_n$$

다음과 같은 논리합 조건을 가진 조인:

$$r \bowtie_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n} s$$

nested loops/block nested loops 조인 방식을 사용하거나,

개별 조인들 $r \bowtie_{\theta i} s$ 내 레코드들의 합집합으로서 계산한다 :

$$(r \bowtie_{\theta 1} s) \cup (r \bowtie_{\theta 2} s) \cup \ldots \cup (r \bowtie_{\theta n} s)$$

기타 연산

중복 제거(**Duplicate elimination**)는 해쉥 또는 정렬을 통해 수행될 수 있다.

- 정렬시 중복은 서로 인접해 있을 것이고, 중복 집합중의 하나만 제외하고는 삭제될 수 있다.
- 최적화 : 중복은 외부 정렬 – 합병에서의 중간 합병 단계에서와

  마찬가지로 실행 파일 생성시 삭제될 수 있다.
- 해쉥도 유사하다 – 중복은 같은 버켓에 있게 될 것이다.

추출(**Projection**)은 각 튜플에 추출을 수행한 후 중복을 제거함으로써 구현된다.

다른 연산들

Aggregation

Set operations ($\cup$, $\cap$ and $-$)

Outer join

표현식의 평가

지금까지: 각 개별 연산을 위한 알고리즘을 검토하였다.

전체 표현식 트리를 평가하는 방법

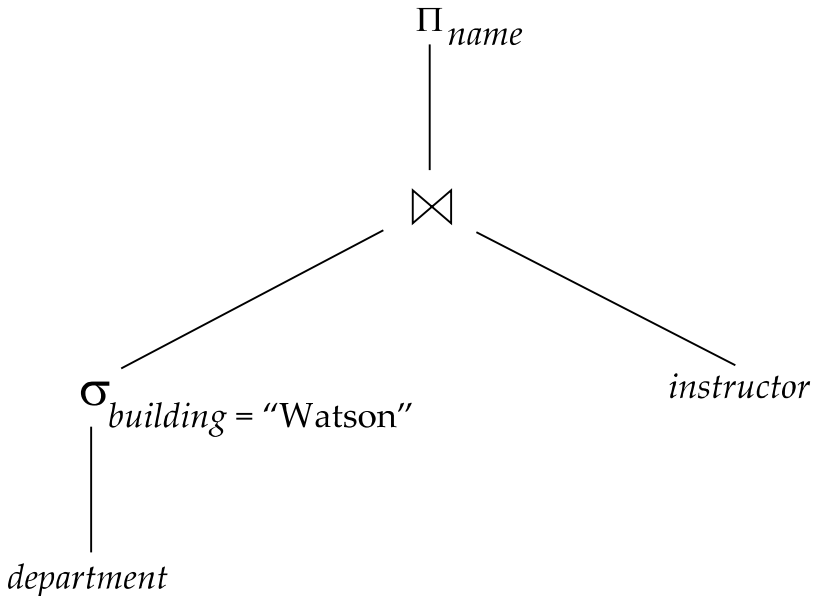
구체화(Materialization):

파이프라이닝 (Pipelining):

구체화 (Materialization)

구체화 : 최하위 단계에서 시작해 한 번에 하나의 연산을 평가한다. 임시 릴레이션에 구체화시킨 중간 결과를 이용하여 다음 단계 연산을 수행한다.

아래 그림의 예에서, $\sigma_{building="Watson"}(department)$ 를 계산하고 저장한다. 다음, *instructor* 와 죠인하고 저장하고, 최종적으로 name 추출 연산을 수행한다.



파이프라이닝 (Pipelining)

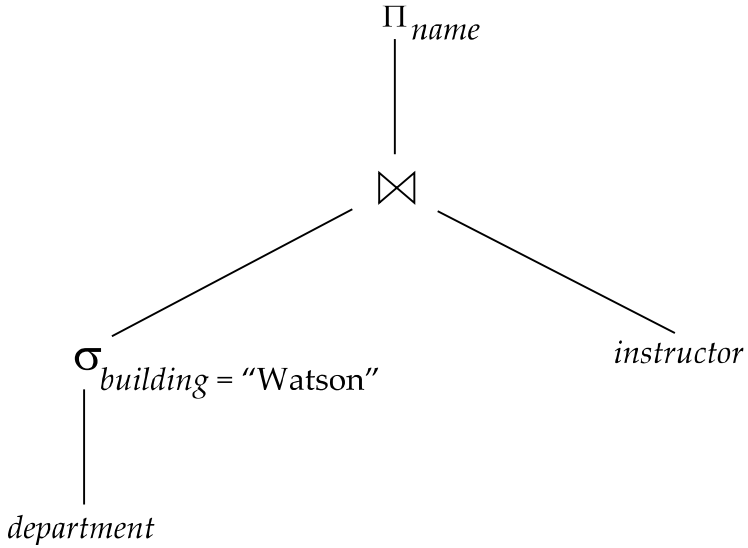
파이프라이닝 : 한 연산의 결과를 다음으로 넘기면서 여러 연산을 동시에 평가한다.
앞의 예에서 다음 연산의 결과를 저장하지 않는다.

$$\sigma_{building="\text{Watson}"}(department)$$

그대신, 튜플들을 직접 죠인 연산 단계로 넘긴다. 이와 유사하게, 죠인 결과도 저장하지 않고 튜플들을 직접 추출 연산으로 넘긴다.

구체화보다 비용이 덜 든다 : 디스크에 임시 릴레이션을 저장할 필요가 없다.

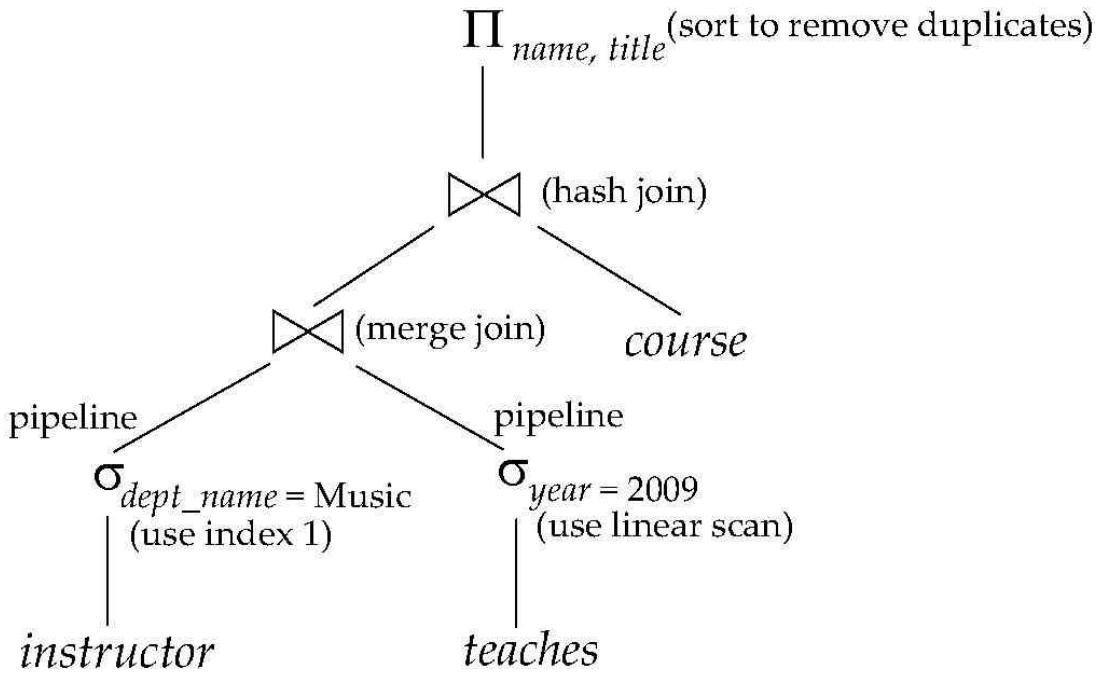
파이프라이닝 분석이 항상 가능한 것은 아니다.



최적화 – 실행 계획 출력

```
SELECT name, title
FROM instructor NATURAL JOIN teaches NATURAL JOIN course
WHERE instructor.dept_name='Music' and year=2009;
```

$\Pi_{name,\ title}(\sigma_{dept\_name=\text{"Music"} \wedge year = 2009}\ (instructor \bowtie (teaches \bowtie (course))))$



End of Chapter