Chapter 13: 질의 최적화

Database System Concepts, 6th Ed.

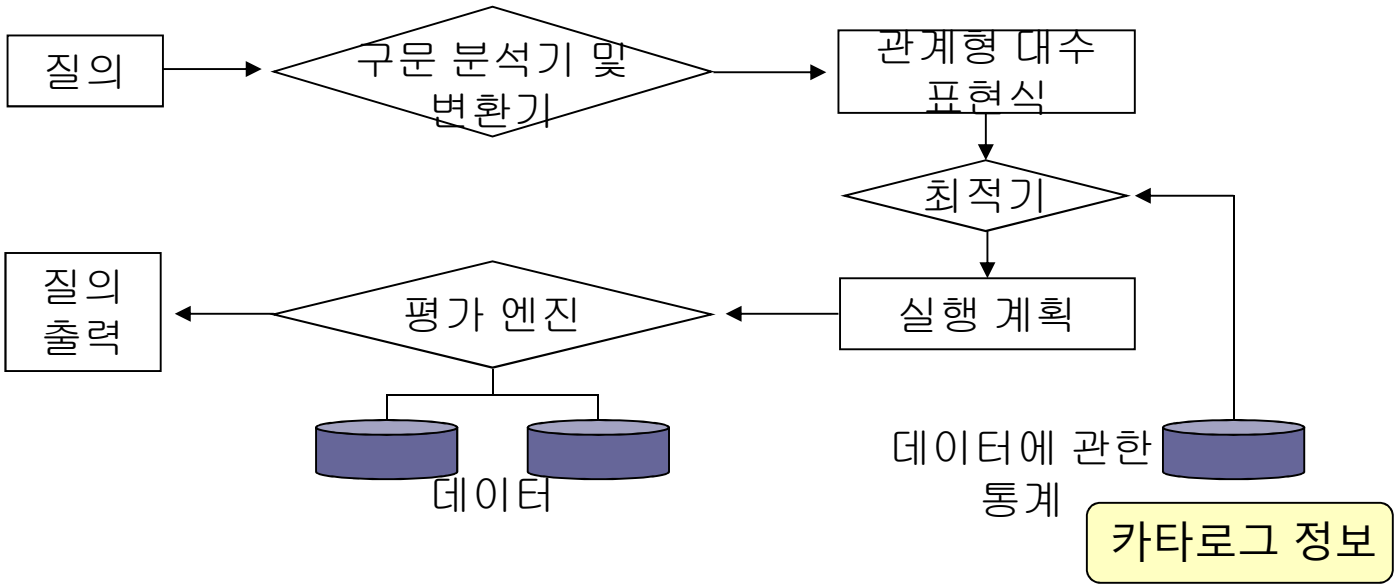
Chapter 13: 질의 최적화

개요

관계형 표현식의 변환

비용 기반 최적화

최적화 절차



개요

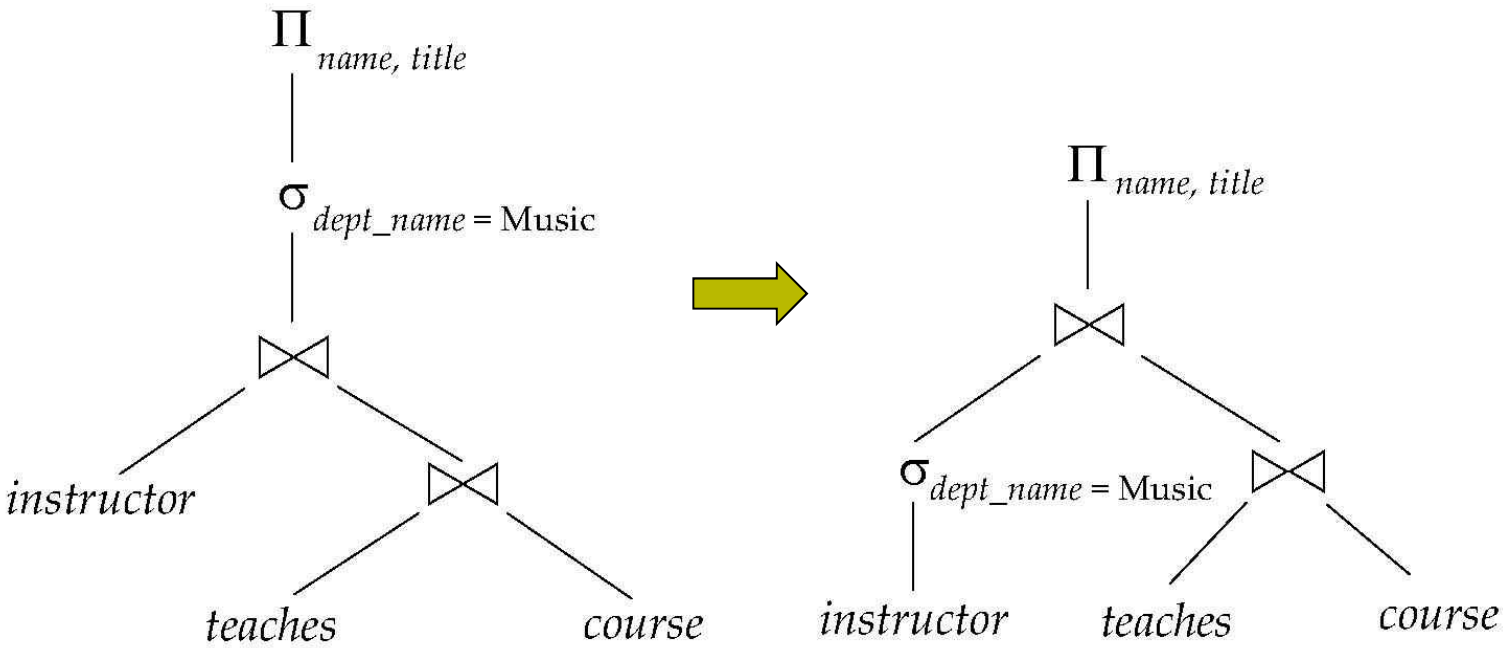
주어진 질의를 평가하는 방법

동일한 표현식으로 바꾸어 평가하기

각 연산에 서로 다른 알고리즘을 적용하기

```
SELECT name, title
FROM instructor NATURAL JOIN teaches NATURAL JOIN course
WHERE instructor.dept_name="Music";
```

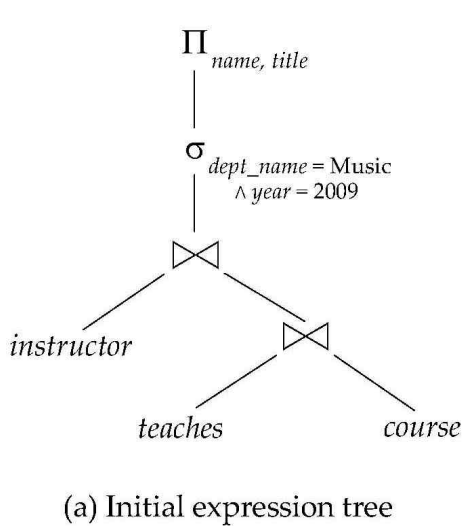
$\Pi_{name,\ title}(\sigma_{dept\_name=\text{"Music"}}\ (instructor \bowtie (teaches \bowtie (course))))$



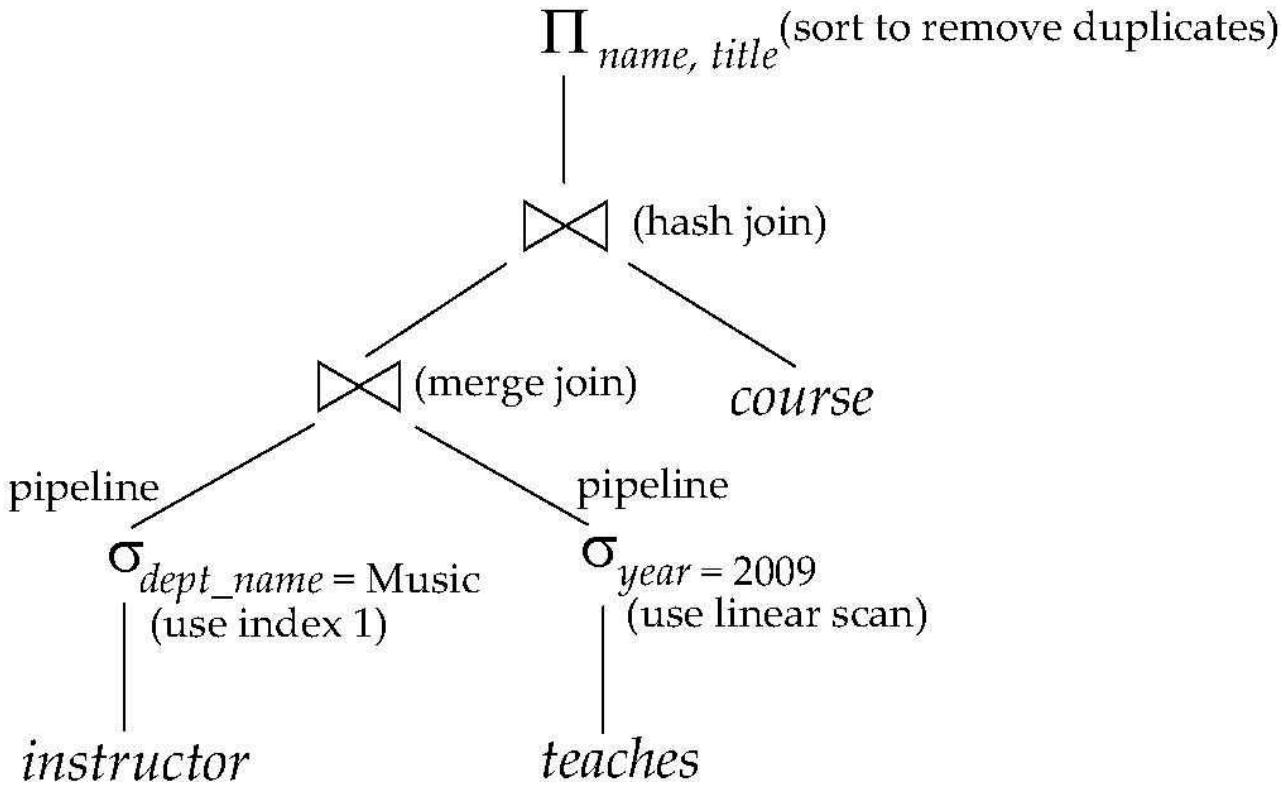
개요 (계속)

평가 계획 (**evaluation plan**)은 각 연산에 어떤 알고리즘이 사용되고, 또한 각 연산이 어떤 순서로 실행되는 지의 절차 등을 정의한다.

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



(a) Initial expression tree



개요 (계속)

주어진 질의에 대한 서로 다른 평가 계획의 비용 차는 매우 클 수 있다.

E.g. seconds vs. days in some cases

비용 기반 최적화 단계

1. 동등 규칙 (**equivalence rules**)을 사용하여 논리적으로 동등한 표현식을 생성한다.
2. 결과 표현식에 주석을 달아 다수의 평가 계획을 얻는다.
3. 산정 비용 (**estimated cost**)에 근거해 가장 비용이 싼 계획을 선택한다.

다음 사항에 근거하여 각 계획의 비용을 산정한다:

릴레이션의 통계적 정보

- 튜플 수, 각 속성의 서로 다른 값의 수 등

중간 결과에 대한 통계적 추정 값

사용 알고리즘에 대한 비용 등

동등 표현식 생성 방법

Database System Concepts, 6th Ed.

동등성 규칙 (Equivalence Rules)

1. 논리곱 선택 연산은 일련의 개별 선택으로 분리될 수 있다.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. 선택 연산은 교환 법칙이 성립한다

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. 일련의 추출 연산들중 마지막 것만이 필요하고, 다른 것들은 생략될 수 있다.

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L_1}(E)$$

4. 선택은 카티전 곱과 세타 죠인으로 결합될 수 있다.
   a. $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
   b. $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

동등성 규칙 (계속)

5. 세타 죠인 연산 (및 자연 죠인)은 교환 법칙이 성립한다.

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$

6. (a) 자연 죠인 연산은 결합 법칙이 성립한다:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

(b) 세타 죠인은 다음과 같은 방식의 결합 법칙이 성립한다:

$$(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$$

여기에서 θ_2 는 E_2 과 E_3 로부터의 애트리뷰트만을 내포한다.

동등석 규칙 (계속)

7. 선택 연산은 다음과 같은 두 조건하에서 세타 죠인 연산에 걸쳐 분배된다 :

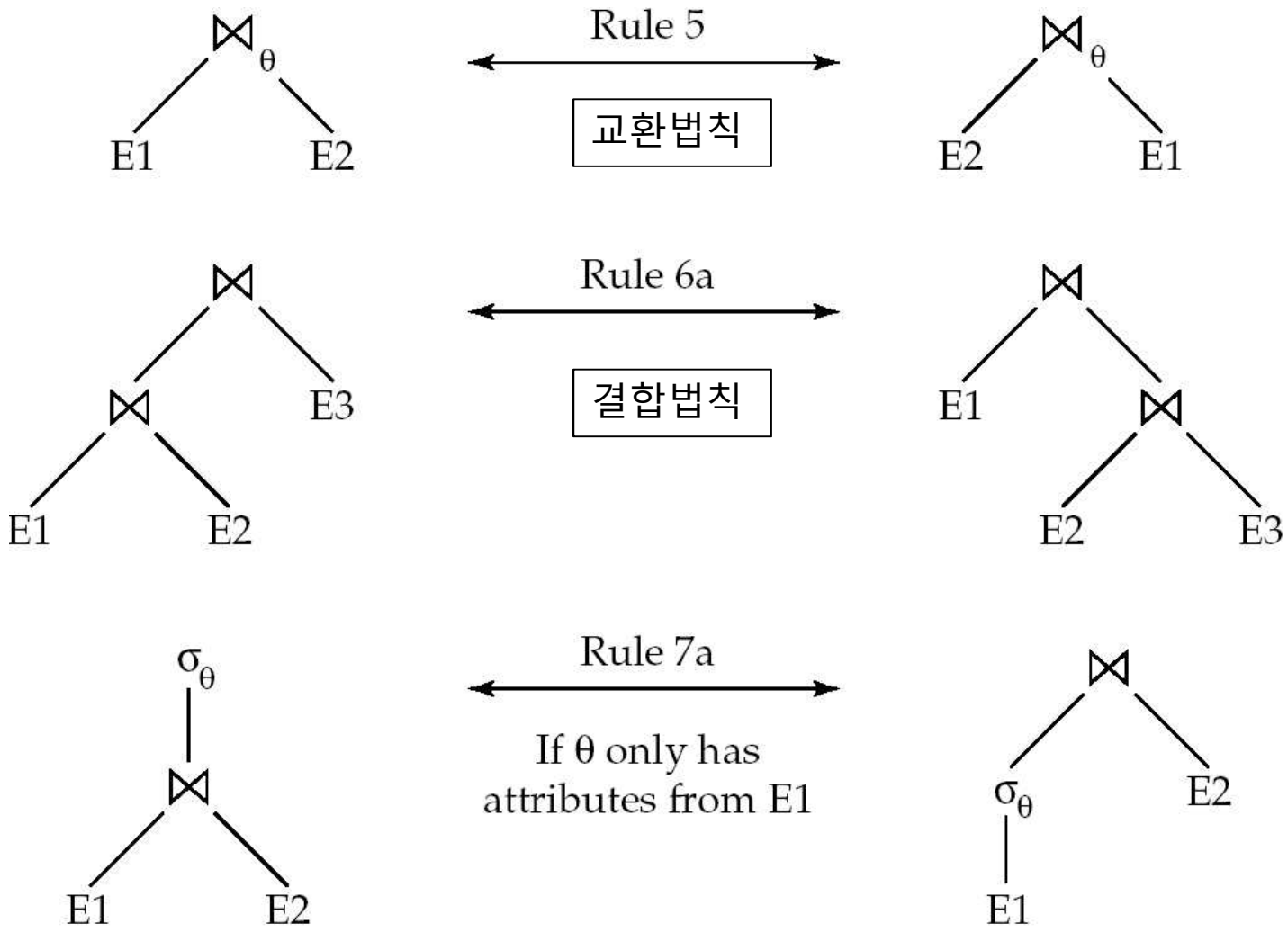
(a) θ_0내 모든 애트리뷰트가 죠인되고 있는 표현식 중의 하나(E_1)의 애트리뷰트만을 내포할 때:

$$\sigma_{\theta 0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_\theta E_2$$

(b) θ_1은 E_1의 애트리뷰트만을 내포하고 θ_2는 E_2의 애트리뷰트만을 내포할 때:

$$\sigma_{\theta 1 \wedge \theta 2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta 1}(E_1)) \bowtie_\theta (\sigma_{\theta 2}(E_2))$$

동등성 규칙 (계속)



동등 규칙 (계속)

8. 추출 연산은 다음과 같이 세타 조인 연산에 걸쳐 분배된다 :

(a) θ가 $L_1 \cup L_2$의 애트리뷰트만을 내포하면

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$$

(b) $E_1 \bowtie_\theta E_2$ 조인을 고려해 보자.

- L_1과 L_2를 각각 E_1과 E_2의 애트리뷰트 집합이라 하자.
- L_3는 조인 조건 θ에 내포되지만 $L_1 \cup L_2$에는 없는 E_1의 애트리뷰트라 하자.
- L_4는 조인 조건 θ에 내포되지만 $L_1 \cup L_2$에는 없는 E_2의 트리뷰트라 하자.

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\Pi_{L_2 \cup L_4}(E_2)))$$

동등 규칙 (계속)

9. 합집합과 공통 집합 연산은 교환 법칙이 성립한다

$$E_1 \cup E_2 = E_2 \cup E_1$$
$$E_1 \cap E_2 = E_2 \cap E_1$$

(차집합 연산은 교환 법칙이 성립하지 않는다).

10. 합집합과 공통 집합은 결합 법칙이 성립한다 .

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$
$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

11. 선택 연산은 $\cup$, $\cap$ 및 – 에 걸쳐 분배된다.

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - \sigma_\theta(E_2)$$

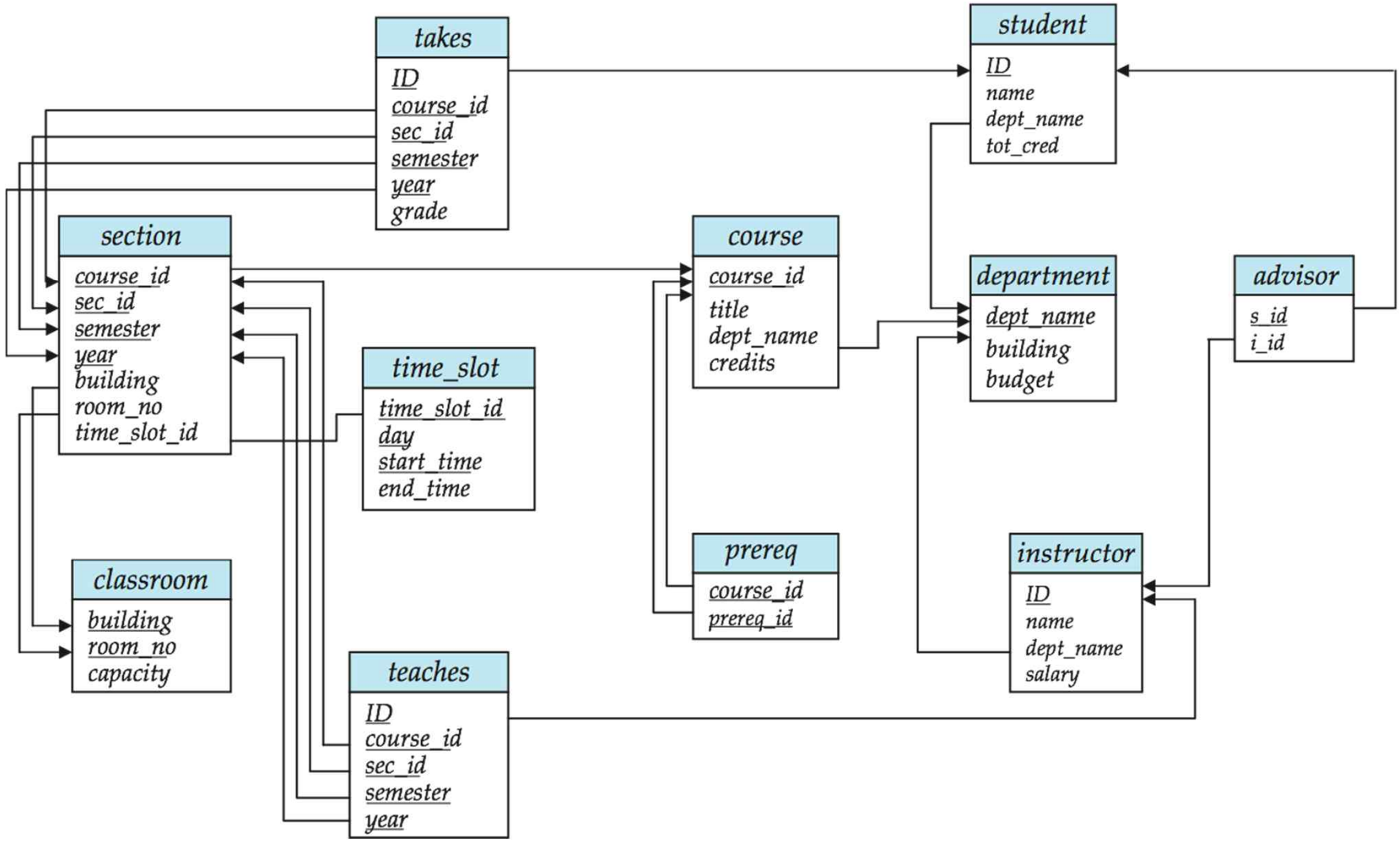
차집합과 공통 집합에 대해서는 다음이 성립한다.

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta(E_1) - E_2$$

12. 추출 연산은 합집합 연산에 걸쳐 분배된다.

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

Schema Diagram for University Database



변환 예제: 선택연산

instructor(**ID**, name, dept_name, salary)
teaches(**ID, course_id, sec_id, semester, year**)
course(**course_id**, title, dept_name, credits)

질의: Music department 에 소속된 모든 강사 이름과 그 들이 가르친 코스 명을 찾아라.

$\Pi_{name,\ title}(\sigma_{dept\_name=\text{“Music”}}(instructor \bowtie (teaches \bowtie course)))$

동등 조건 규칙 7a 를 사용한 변환

> 7. 선택 연산은 다음과 같은 두 조건하에서 세타 죠인 연산에 걸쳐 분배된다 :
> (a) θ_0내 모든 애트리뷰트가 죠인되고 있는 표현식 중의 하나(E_1)의 애트리뷰트만을 내포할 때:
>
> $$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$

$\Pi_{name,\ title}((\sigma_{dept\_name=\text{“Music”}}(instructor)) \bowtie (teaches \bowtie course))$

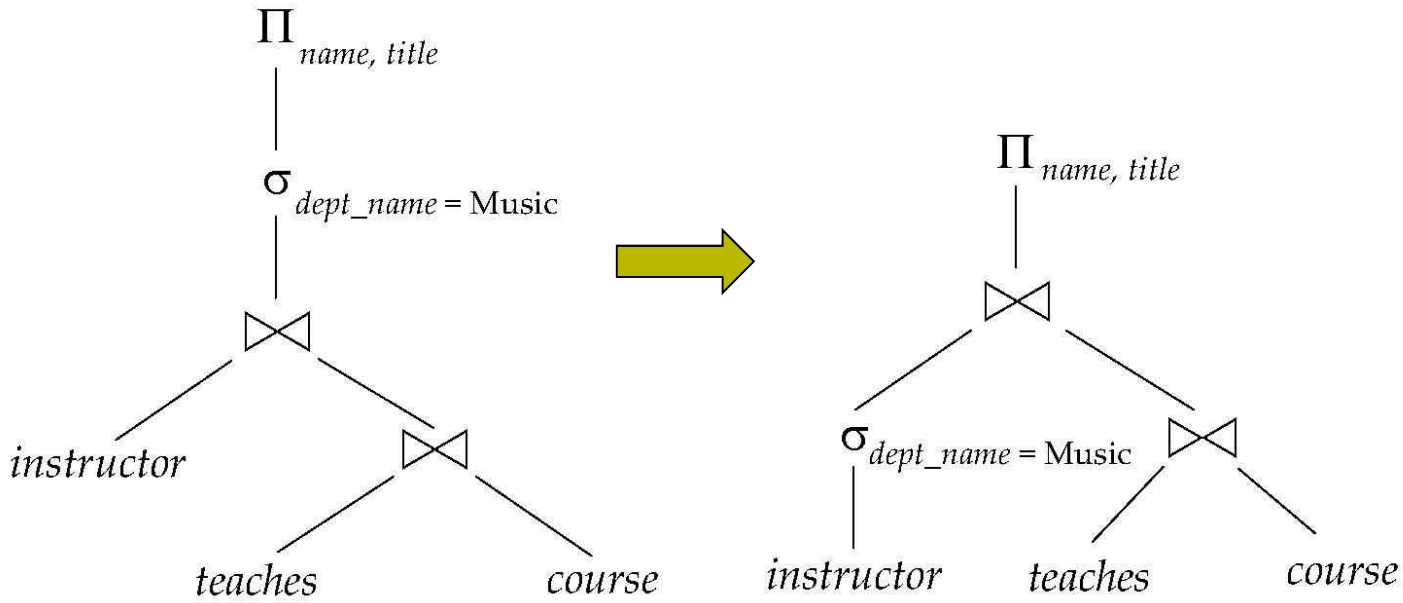
선택 연산을 가능한 범위에서 먼저 수행하면 죠인될 릴레이션의 크기를 줄일 수 있다.

변환 예제: 선택 연산을 가능한 범위에서 먼저 수행하면 조인될 릴레이션의 크기를 줄일 수 있다

질의: Music department 에 소속된 모든 강사 이름과 그 들이 가르친 코스 명을 찾아라.

$\Pi_{name,\ title}(\sigma_{dept\_name=\text{“Music”}}(instructor \bowtie (teaches \bowtie course)))$

$\Pi_{name,\ title}((\sigma_{dept\_name=\text{“Music”}}(instructor)) \bowtie (teaches \bowtie course))$



선택 연산 예제 (계속)

instructor(**ID**, name, dept_name, salary)
teaches(**ID, course_id, sec_id, semester, year**)
course(**course_id**, title, dept_name, credits)

질의: 2009년에 임의의 코스를 가르친 Music department 에 소속된 모든 강사 이름과 그 들이 가르친 코스 명을 찾아라.

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

동등성 규칙 중 조인의 결합 법칙을 사용한 변환 (규칙 6a) :

> 6. (a) 자연 조인 연산은 결합 법칙이 성립한다:
> $(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$

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

두번째 형식에 “가능한 먼저 선택” 규칙을 적용해 다음과 같은 부 표현식을 얻는다

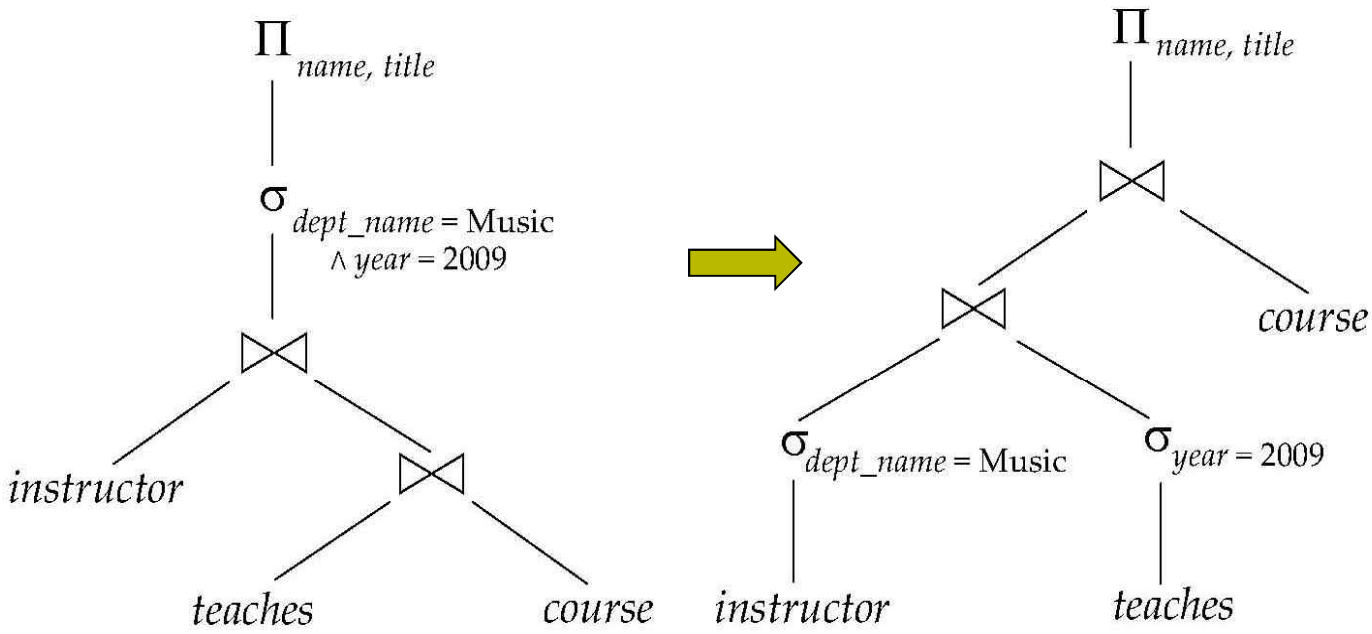
$\Pi_{name,\ title}(\sigma_{dept\_name=\text{“Music”} \wedge year = 2009}((instructor \bowtie teaches) \bowtie course))$

$\sigma_{dept\_name = \text{“Music”}}(instructor) \bowtie \sigma_{year = 2009}(teaches)$

변환 예제 (계속)

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

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



(a) Initial expression tree

(b) Tree after multiple transformations

변환 예제: 추출 연산

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

instructor(**ID**, name, dept_name, salary)
teaches(**ID, course_id, sec_id, semester, year**)
course(**course_id**, title, dept_name, credits)

다음을 계산하면

$(\sigma_{dept\_name = \text{“Music”}}(instructor) \bowtie teaches)$

아래와 같은 스키마를 가진 릴레이션을 얻는다:
(*ID, name, dept_name, salary,* ***course_id, sec_id, sem, year)***

동등성 규칙 8a와 8b를 사용해 추출을 앞으로 밀어내 얻어지는 중간 결과에서 불필요한 애트리뷰트를 제거한다 :

θ가 $L_1 \cup L_2$의 애트리뷰트만을 내포하면

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$$

$$\Pi_{name,\ title}((\Pi_{name,\ course\_id}(\sigma_{dept\_name=\text{“Music”}}(instructor) \bowtie teaches)) \bowtie \Pi_{course\_id,\ title}(course))$$

추출 연산을 가능한 범위에서 먼저 수행하면 죠인될 릴레이션의 크기를 줄일 수 있다.

죠인 순서화 예제

모든 릴레이션 r_1, r_2 및 r_3에 대해

$$(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$$

(동등성 규칙 6. Join의 결합법칙)

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

$r_2 \bowtie r_3$가 매우 크고 $r_1 \bowtie r_2$가 작다면, 다음과 같이 선택하여

$$(r_1 \bowtie r_2) \bowtie r_3$$

보다 작은 임시 릴레이션을 계산하고 저장하도록 한다.

죠인 순서화 예제 (계속)

다음 표현식을 고려해 보자

$$\Pi_{name,\ title}((\sigma_{dept\_name=\ \text{“Music”}}(instructor) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course))$$

teaches $\bowtie \Pi_{course\_id,\ title}$ (*course*) 를 먼저 계산하고, 그 후에

$\sigma_{dept\_name=\ \text{“Music”}}$ (*instructor*) 와 조인할 수 있다.

그러나, 처음 연산의 죠인 릴레이션은 큰 릴레이션일 가능성이 있다.

대학 강사 중 소수만이 Music department 에 소속되어 있을 가능성이 보다 크므로, 다음을 먼저 계산하는 것이 보다 낫다.

$$\sigma_{dept\_name=\ \text{“Music”}}(instructor) \bowtie teaches$$

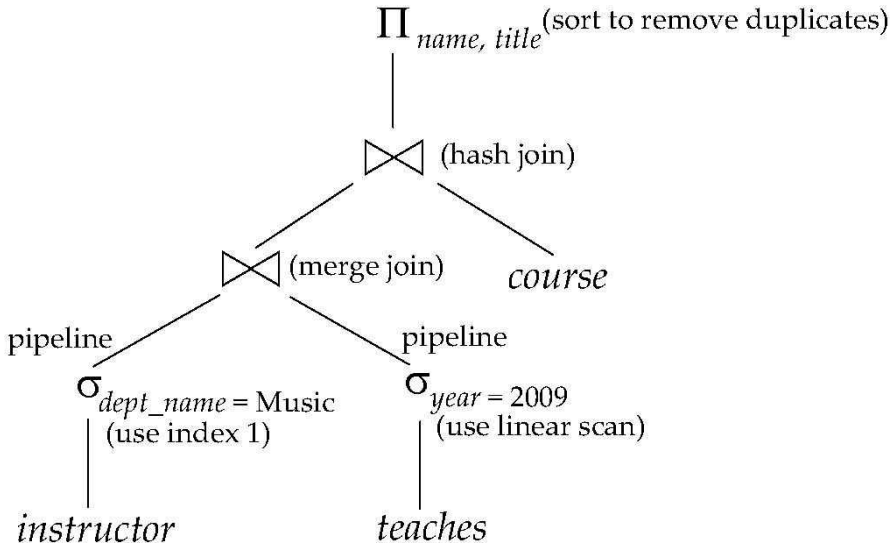
평가 계획의 선택

평가 계획(evaluation plan)을 선택할 때는 평가기법의 상호 작용을 고려해야 한다 : 각 연산에 대해 독립적으로 가장 비용이 적은 알고리즘을 선택하는 것이 전체적으로 최상의 알고리즘을 생성하지 못할 수도 있다. 예를 들어,

- 합병–죠인이 해쉬–죠인보다 비용이 더 들 수 있지만, 외부 단계의 집성 연산에 대해서는 비용을 줄이는 정렬된 출력을 제공할 수도 있다.
- 내포 –루프 죠인은 파이프라이닝을 위한 기회를 제공한다.

실제 질의 최적기는 다음과 같은 두 가지 넓은 방법들의 요소를 결합한다 :

1. 모든 계획을 검색해 비용기반 유형에서 최상의 계획을 선택한다.
2. 경험을 사용해 계획을 선택한다.



비용 기반 최적화

$r_1 \bowtie r_2 \quad \ldots \bowtie r_n$에 대한 최상의 죠인 순서를 찾아 보자.

위의 표현식에 대해 (2(n - 1))! / (n - 1)!의 죠인 순서가 존재한다. n = 7이면 그 수는 665280이고, n = 10이면 그 수는 1760억 보다 크다.

모든 죠인 순서를 생성할 필요가 없다. 동적 프로그래밍을 사용하면,

$\{r_1, r_2, \ldots, r_n\}$의 어떤 부분 집합에 대한 최소 비용의 죠인 순서는 한번만 계산되고 미래의 사용을 위해 저장된다.

이것은 시간 복잡성을 대략 $O(3^n)$으로 줄인다. n = 10이면, 이 수는 59000이다.

최적화에서의 동적 프로그래밍(생략)

n 릴레이션 집합에 대한 최상의 왼쪽–깊이 죠인 트리를 찾으려면 :

- 하나의 릴레이션을 오른쪽 편 입력으로 하고 다른 릴레이션들은 왼쪽 편 입력으로 한 n개의 대안을 고려한다.
- 왼쪽 편의 각 대안에 대해 최소 비용 죠인 순서를 사용해 (반복해 계산하고 저장함), n개 대안중 가장 비용이 적은 것을 선택한다.

n 릴레이션 집합에 대한 최상의 죠인 트리를 찾으려면 :

- n 릴레이션중 집합 S에 대한 최상의 계획을 찾기 위해, 다음과 같은 형식의 모든 가능한 계획을 고려한다 : $S_1 \bowtie (S - S_1)$, 여기서 S_1은 S의 어떤 공집합이 아닌 부분 집합.
- 이전처럼 각 계획의 비용을 찾기 위해 S의 부분 집합에 대해 반복적으로 계산하고 저장된 비용을 사용한다.

$2^n - 1$ 대안중 가장 비용이 적은 것을 찾는다.

비용 기반 최적화에서 흥미있는 순서 (생략)

표현식 $(r_1 \bowtie r_2 \bowtie r_3) \bowtie r_4 \bowtie r_5$ 를 고려해 보자.

흥미있는 정렬 순서란 이후의 연산에 사용될 수 있는 튜플들의 특정 정렬 순서이다.

- r_4 또는 r_5에 공통인 애트리뷰트로 정렬된 $r_1 \bowtie r_2 \bowtie r_3$ 의 결과를 생성하는 것은 유용하지만, r_1과 r_2에 공통인 애트리뷰트로 생성하는 것은 유용하지 않다.
- 합병–죠인을 사용해 $r_1 \bowtie r_2 \bowtie r_3$ 를 계산하는 것이 비용이 더 들지만, 흥미있는 순서로 정렬된 출력을 제공한다.

주어진 n 릴레이션 집합 각 부분 집합에 대해 최상의 죠인 순서를 찾는 것으로는 불충분하다 ; 그 부분 집합을 위한 죠인 결과의 각 흥미있는 정렬 순서에 대해 각 부분 집합에 대한 최상의 죠인 순서를 찾아야 한다. 앞의 동적 프로그래밍 알고리즘을 단순히 확장하면 된다.

경험적 최적화

동적 프로그래밍을 도입하더라도 비용 기반 최적화는 경비가 많이 든다.

시스템은 비용 기반 유형으로 이루어져야 하는 선택의 수를 줄이는데 경험을 사용할 수 있다.

경험적 최적화는 일반적으로 실행 성능을 향상시키는 (모든 경우는 아니지만) 규칙의 집합을 사용해 질의 트리를 변환한다.

- 가능한 먼저 선택을 수행한다 (튜플의 수를 줄인다).
- 가능한 먼저 추출을 수행한다 (애트리뷰트의 수를 줄인다).
- 다른 유사한 연산 이전에 가장 제한적인 선택과 조인 연산을 수행한다.

어떤 시스템에서는 경험만을 사용하고, 다른 시스템에서는 경험과 부분적인 비용 기반 최적화를 결합한다.

전형적인 최적화 절차

1. 논리곱 선택을 일련의 단일 선택 연산으로 분해한다 (규칙 1).

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. 가장 먼저 실행이 되도록 선택 연산을 질의 트리 아래로 이동 시킨다 (규칙 2, 7a, 7b, 11).

$$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$

3.가장 작은 릴레이션을 생성할 선택과 죠인 연산을 먼저 실행한다 (규칙 6).

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

4. 선택 조건이 뒤에 나오는 카티전 곱 연산을 죠인 연산으로 대치한다 (규칙 4a)

$$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$

5. 필요한 곳에 새로운 추출을 생성하면서 추출 애트리뷰트 리스트를 분해해 가능한 트리의 아래로 이동시킨다 (규칙 3, 8a, 8b, 12)

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = (\Pi_{L_1}(E_1)) \bowtie_{\theta} (\Pi_{L_2}(E_2))$$

6. 연산들이 파이프라인될 수 있는 부 트리를 인지하여 파이프라이닝을 사용해 실행한다.

End of Chapter