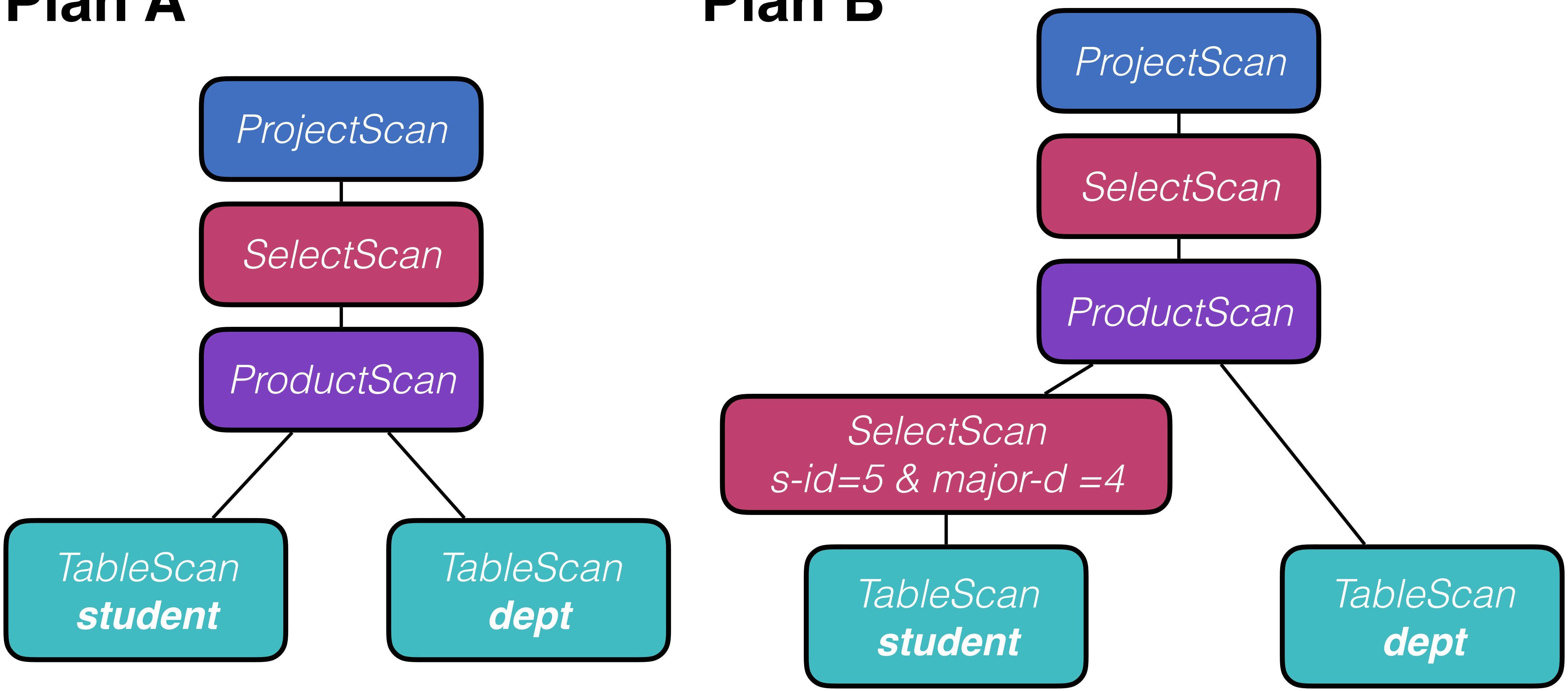
Midterm Project

ctsai@DataLAB
Cloud Database
2017 Spring

Revisit Query Processing

```
SELECT sname FROM student, dept
        WHERE major-id = d-id
              AND s-id = 5 AND major-id = 4;
```

Plan A **Plan B**



Revisit Query Processing

- A good scan tree can be faster than a bad one for orders of magnitude
- Consider the product scan at middle
- Let Row(student)=10000, Block(student)=1000, Block(dept)= 500, and selectivity(s-id=5&major-id=4)=0.01
- Each block access requires 10ms
- Cost:
    - Plan A : (10000 + 10000*500)*10ms = 13.9 hours
    - Plan B : (10000 + 10000*0.001*500)*10ms = 8.4min

Query Optimization

- Finding a good plan is crucial
- Cost difference between evaluation plans for a query can be enormous
  - E.g. seconds vs. days in some cases
- Midterm Project : Implement one of classic query optimizer
- What you will need …
  - Query plan cost estimation
  - Transformation of query plans
  - Some heuristic optimization
  - Selinger optimizer <— you are going to implement it !!

How to say two Relational Expressions are equivalent ?

- Two relational algebra expressions are said to be **equivalent** if the two expressions generate the same set of tuples on every legal database instance
    - Note that the order of tuples is irrelevant
- An **equivalence rule** says that expressions of two forms are equivalent
- Some notations :

ProjectScan	*SelectScan*	*ProductScan*
Π_{L_1}	σ_{θ_1}	$\bowtie$

Equivalence Rules

1. Conjunctive selection operations can be deconstructed into a sequence of individual selections.

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

2. Selection operations are commutative.

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

3. Only the last in a sequence of projection operations is needed, the others can be omitted.

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

4. Selections can be combined with Cartesian products and theta joins.

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

Equivalence Rules

5. Theta-join operations (and natural joins) are commutative

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

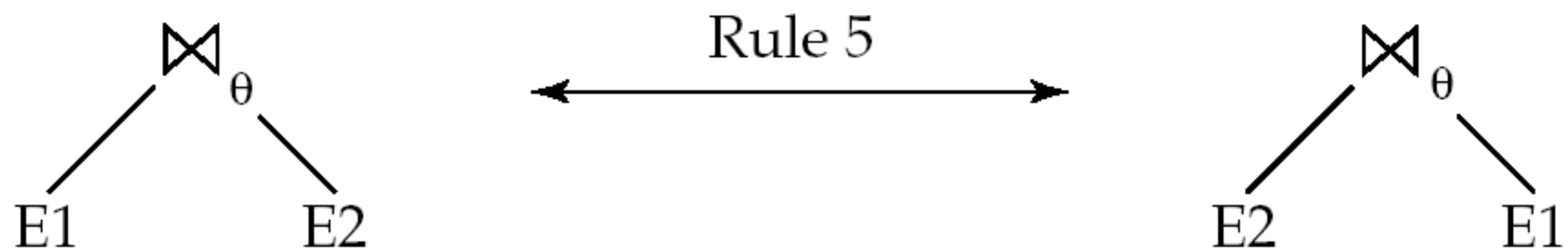
6. Natural join operations are associative:

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

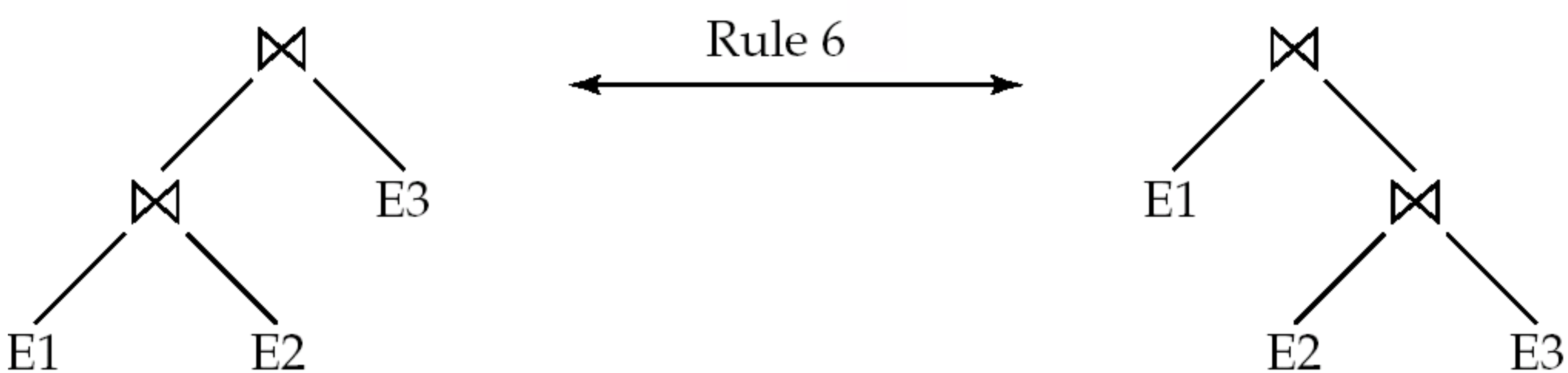
7. The selection operation distributes over the theta join operation

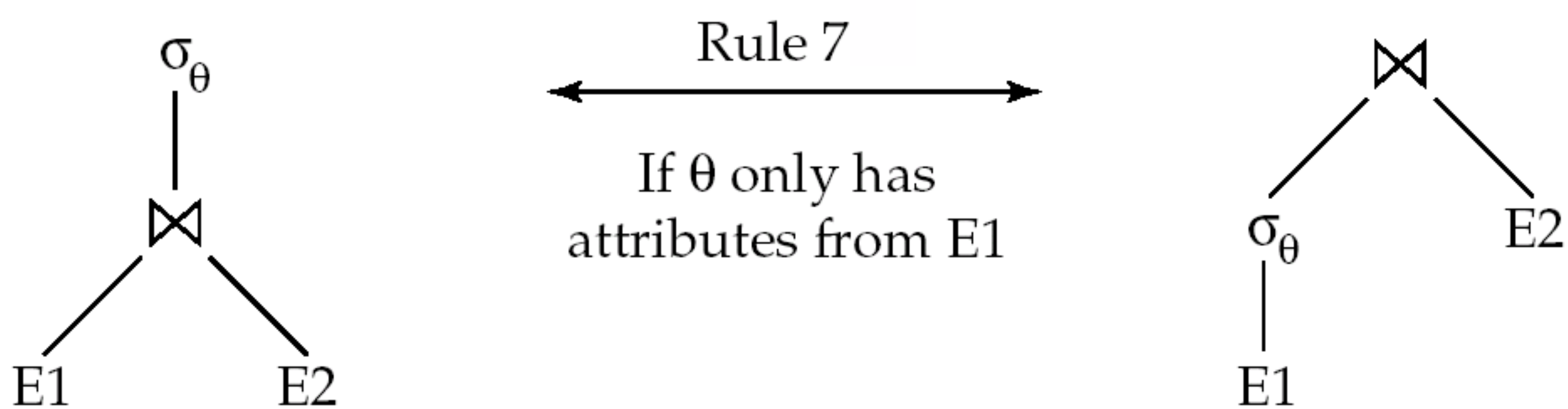
$$\sigma_{\theta 0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_\theta E_2$$

Equivalence Rules



Inner or outer product





Pushing Selections

- Query: Find the names of all instructors in the Music department who have taught a course in 2009, along with the titles of the courses that they taught

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

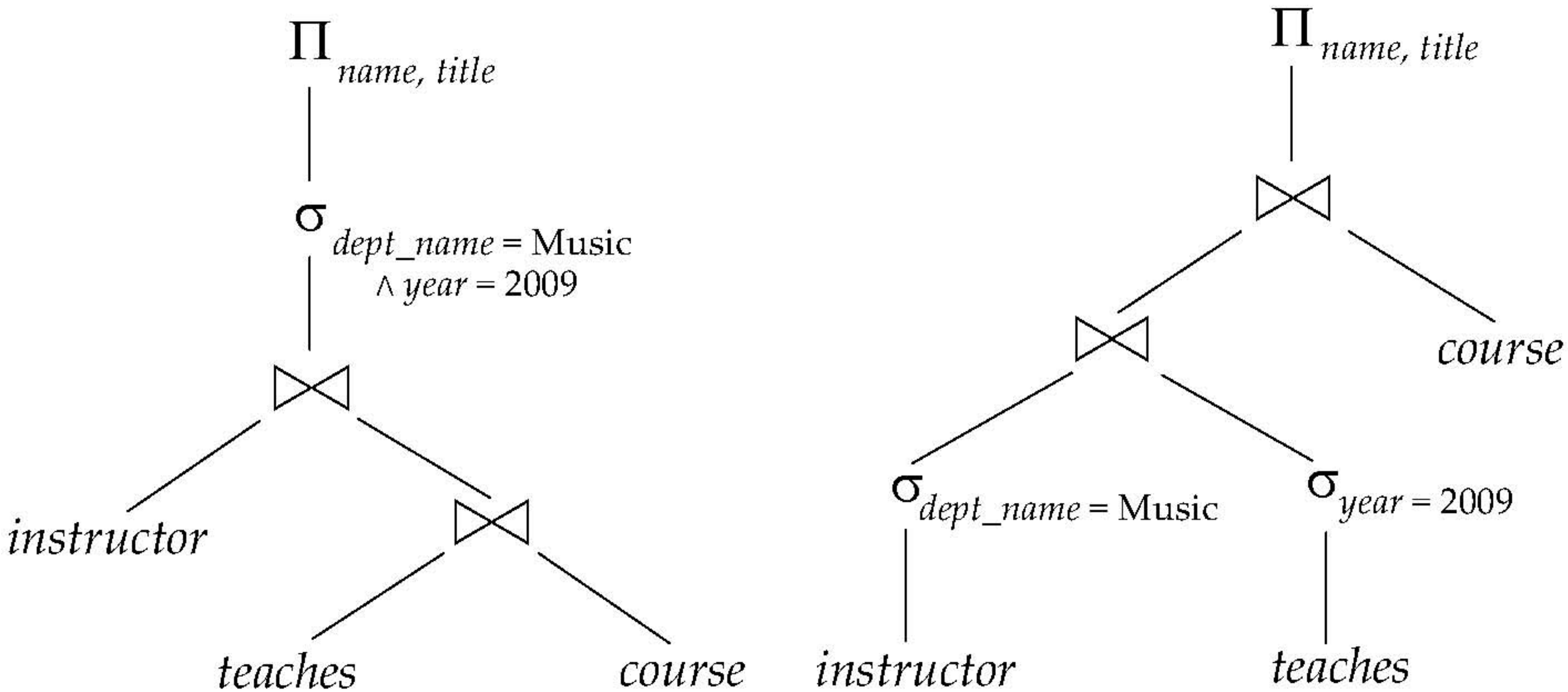
- Transformation using join associatively (Rule 6):

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

- Second form provides an opportunity to apply the "perform selections early" rule, resulting in the subexpression

$$\sigma_{dept\_name = \text{"Music"}}(\ instructor) \quad \sigma_{year = 2009}(\ teaches)$$

Pushing Selections

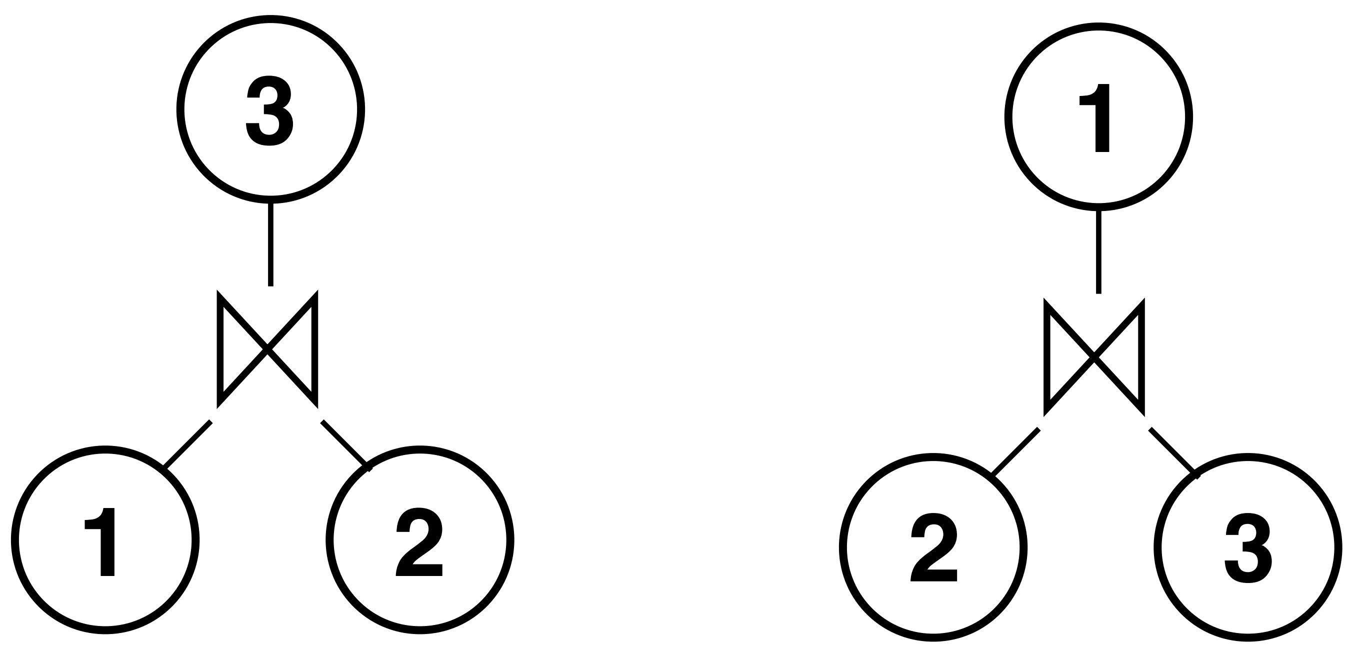


(a) Initial expression tree　　(b) Tree after multiple transformations

- Performing the selection as early as possible reduces the size of the relation to be joined.

Join order problem

- Form previous example you known the order of join table have great effect to scan cost
- Consider finding the best join-order for r1 ⋈ r2 ⋈ . . . rn.
  - {1, 2 , 3} : ((1 2)3), (1(2 3)), ((1 3)2), (1(3 2)) …
  - n!*(n-1)! , e.g. n = 10 , the number is greater 176 billion



((1 2) 3)   (1 (2 3))

Selinger Optimizer

- Access Path Selection in a Relational Database Management System, SIGMOD '79 (Citation : 2412)
- *Selinger Optimizer* was used in System R of IBM and pioneered the idea of cost-based join-order optimization.
- Key implementation decision
  - Left deep tree only (((AB)C)D) (eliminate (n -1)!)
  - Push-down selections
  - Don't consider cross products
  - Dynamic programming algorithm
    - Time : O(n*2^n)

Resource

- Abraham Silberschatz et al, Database System Concepts, 6 Edition, McGraw-Hill, 2010, ISBN: 0073523321
  - Lecture 13
- Database Systems, MIT 6.830 / 6.814
  - Lecture 10 & 11
- Database Internals, CSE 444
  - Query Optimization