Principles of Database Systems (CS307)

Lecture 14: Query Optimization

Zhong-Qiu Wang

Department of Computer Science and Engineering
Southern University of Science and Technology

- Most contents are from slides made by Stéphane Faroult and the authors of Database System Concepts (7th Edition).
- Their original slides have been modified to adapt to the schedule of CS307 at SUSTech.
- The slides are largely based on the slides provided by Dr. Yuxin Ma

Announcements

- Assignment on Trigger, due date: 23:59pm on December 15th 2024, Beijing Time
- For Project II, if you choose to submit
  - by 23:30pm on December 22nd 2024 (Sunday), Beijing time
    - Your project defense has to be in your lab session that is in the same week as the above deadline
    - Your score will be multiplied by 1.1
  - by 23:30pm on December 29th 2024 (Sunday), Beijing time
    - Your project defense has to be in your lab session that is in the same week as the above deadline
    - Your score will be multiplied by 1.0

Query Optimization

- Purpose of query optimization
  - Select an effective way to retrieve the data based on queries while spending the least computational effort
    - However, it is only “spending less computational effort” in most scenarios, not least

Query Optimization

- Users don’t need to consider the best way of writing queries
  - We want DBMS to construct a query-evaluation plan that minimizes the cost of query evaluation
- Automated optimization can perform better (for most of the time)
  - Utilize the data dictionary
  - Real-time utilization based on physical storage changes
  - Optimizer can evaluate hundreds of execution plans in a very short time compared with human programmers
  - Human users do not need to learn advanced optimization techniques any more, which is conducted by optimizers instead

An Example in the Movie Dataset

- The same query can be represented in different plans
  - E.g., retrieve the titles of those movies from China

```sql
select m.title
from movies m, countries c
where m.country = c.country_code and c.country_name = 'China';
```

An Example in the Movie Dataset

- The corresponding relational algebra expressions:

(1) $\Pi_{title}\,(\sigma_{movies.country\,=\,countries.country\_code\ \Lambda\ \text{countries.country}\ =\ \text{“}China\text{”}}(\text{movies} \times \text{countries}))$

(2) $\Pi_{title}\,(\sigma_{\text{countries.country}\ =\ \text{“}China\text{”}}\,(\text{movies} \bowtie_{movies.country\,=\,countries.country\_code} \text{countries}))$

(3) $\Pi_{title}\,(\text{movies} \bowtie_{movies.country\,=\,countries.country\_code} (\sigma_{\text{countries.country}\ =\ \text{“}China\text{”}}\,(\text{countries}))$

An Example in the Movie Dataset

- The corresponding relational algebra expressions:

$$(1)\ \Pi_{title}\ (\sigma_{movies.country\ =\ countries.country\_code\ \Lambda\ \mathrm{countries.country}\ =\ \text{“}China\text{”}}(\mathrm{movies} \times \mathrm{countries}))$$

$$(2)\ \Pi_{title}\ (\sigma_{\mathrm{countries.country}\ =\ \text{“}China\text{”}}\ (\mathrm{movies}\ \bowtie_{movies.country\ =\ countries.country\_code}\ \mathrm{countries}))$$

$$(3)\ \Pi_{title}\ (\mathrm{movies}\ \bowtie_{movies.country\ =\ countries.country\_code}\ (\sigma_{\mathrm{countries.country}\ =\ \text{“}China\text{”}}\ (\mathrm{countries}))$$

  - In (1), a full Cartesian product will be computed, which costs huge time for matching all pairs and massive temporary storage space for the intermediate product table
  - In (2), a smaller intermediate join table is to be cached, which saves some space
  - In (3), the filter ($\sigma_{\mathrm{c.country}\ =\ \text{“}China\text{”}}$) reduces the size of the right table in the join operation, which saves a lot of time for pair matching and caching intermediate join table

An Example in the Movie Dataset

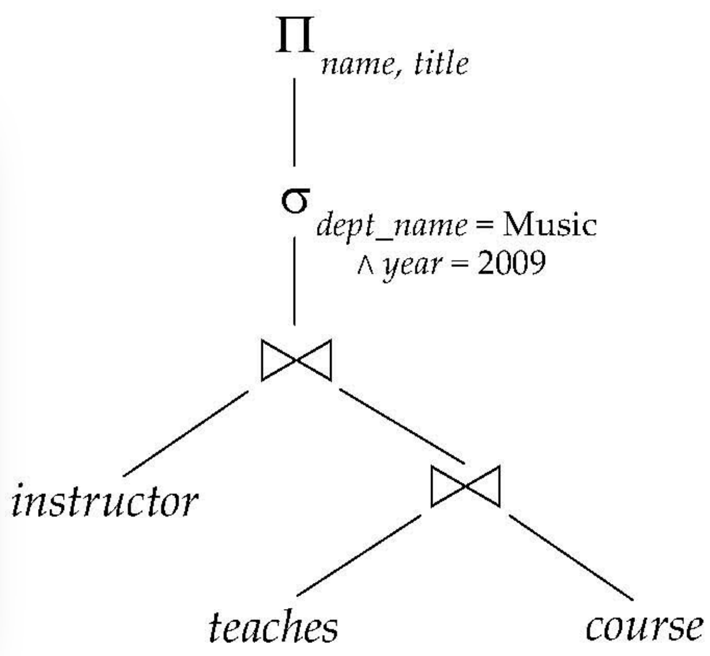
- In addition, the filter operation can be further accelerated once an index is built upon the *country* column

$$(3)\ \Pi_{title}\ (\text{movies} \bowtie_{movies.country\ =\ countries.country\_code} (\sigma_{\text{countries.country}\ =\ \text{“}China\text{”}} (\text{countries})))$$

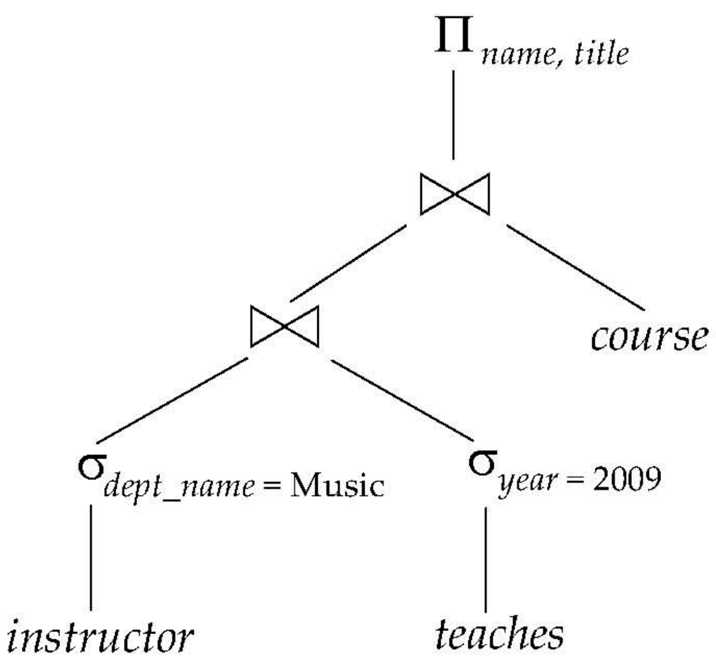
Generating Equivalent Expressions

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation

```sql
select name, title
from instructor
    natural join (teaches natural join course)
where dept_name = 'Music' and year = 2009;
```



(a) Initial expression tree

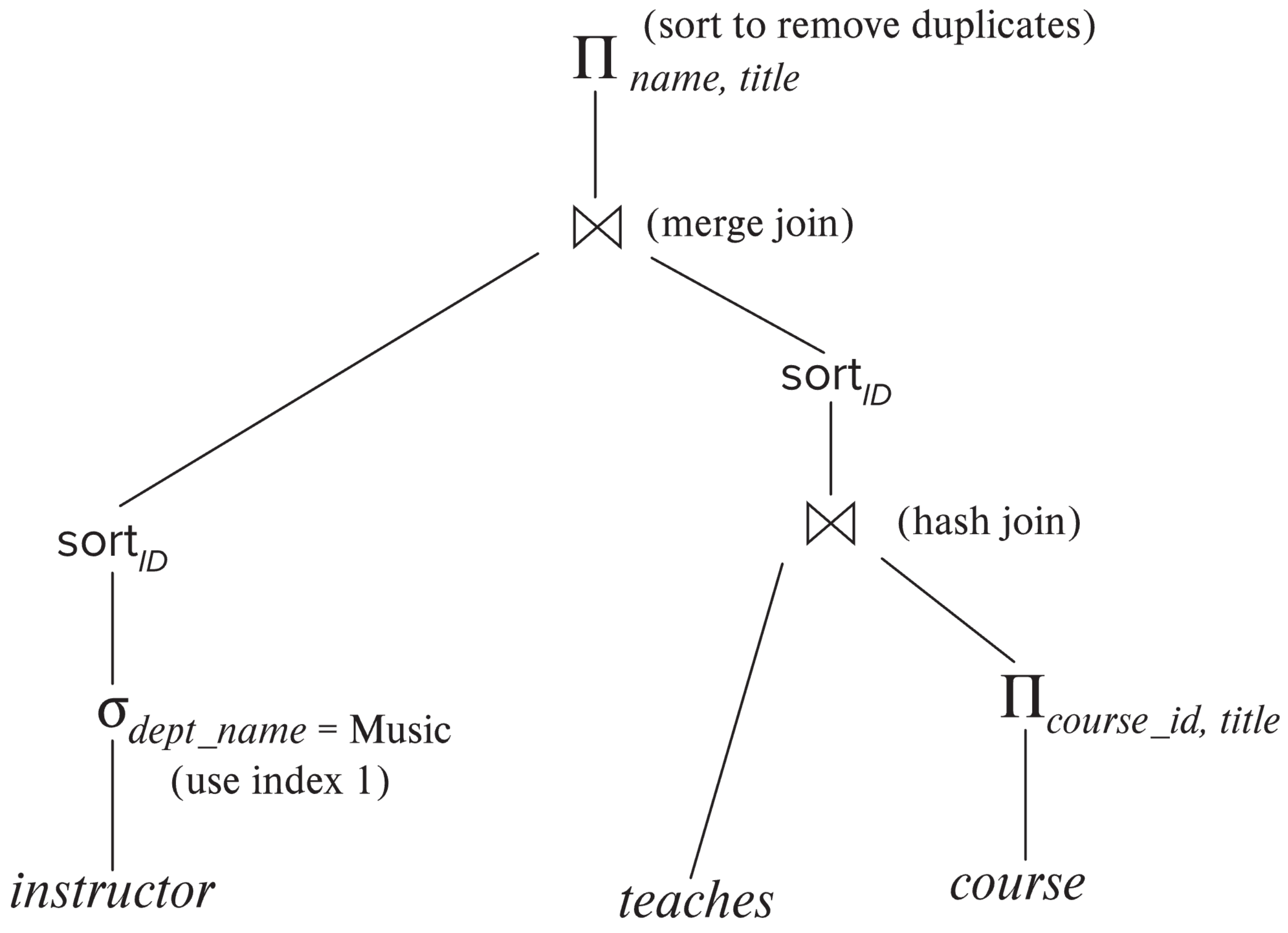


(b) Tree after multiple transformations

Generating Equivalent Expressions

- An **evaluation plan** defines exactly what algorithm is used for each operation, and how the execution of the operations is coordinated

```sql
select name, title
from instructor
    natural join (teaches natural join course)
where dept_name = 'Music' and year = 2009;
```



Transformation of Relational Expressions

- 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: order of tuples is irrelevant
  - We don't care if they generate different results on databases that violate integrity constraints

- An equivalence rule says that expressions of two forms are equivalent
  - ... i.e., we can replace expression of the first form by second, or vice versa
  - The optimizer uses equivalence rules to transform expressions into other logically equivalent expressions

Equivalence Rules

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

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

2. Selection operations are commutative

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

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

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E))\ldots)) \equiv \Pi_{L_1}(E)$$

where $L_1 \subseteq L_2 \ldots \subseteq L_n$

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

a) $\sigma_{\theta}(E_1 \times E_2) \equiv E_1 \bowtie_{\theta} E_2$ (same as the definition of theta-join)

b) $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) \equiv E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

Equivalence Rules

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

$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$$

6. (a) Natural join operations are associative:

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

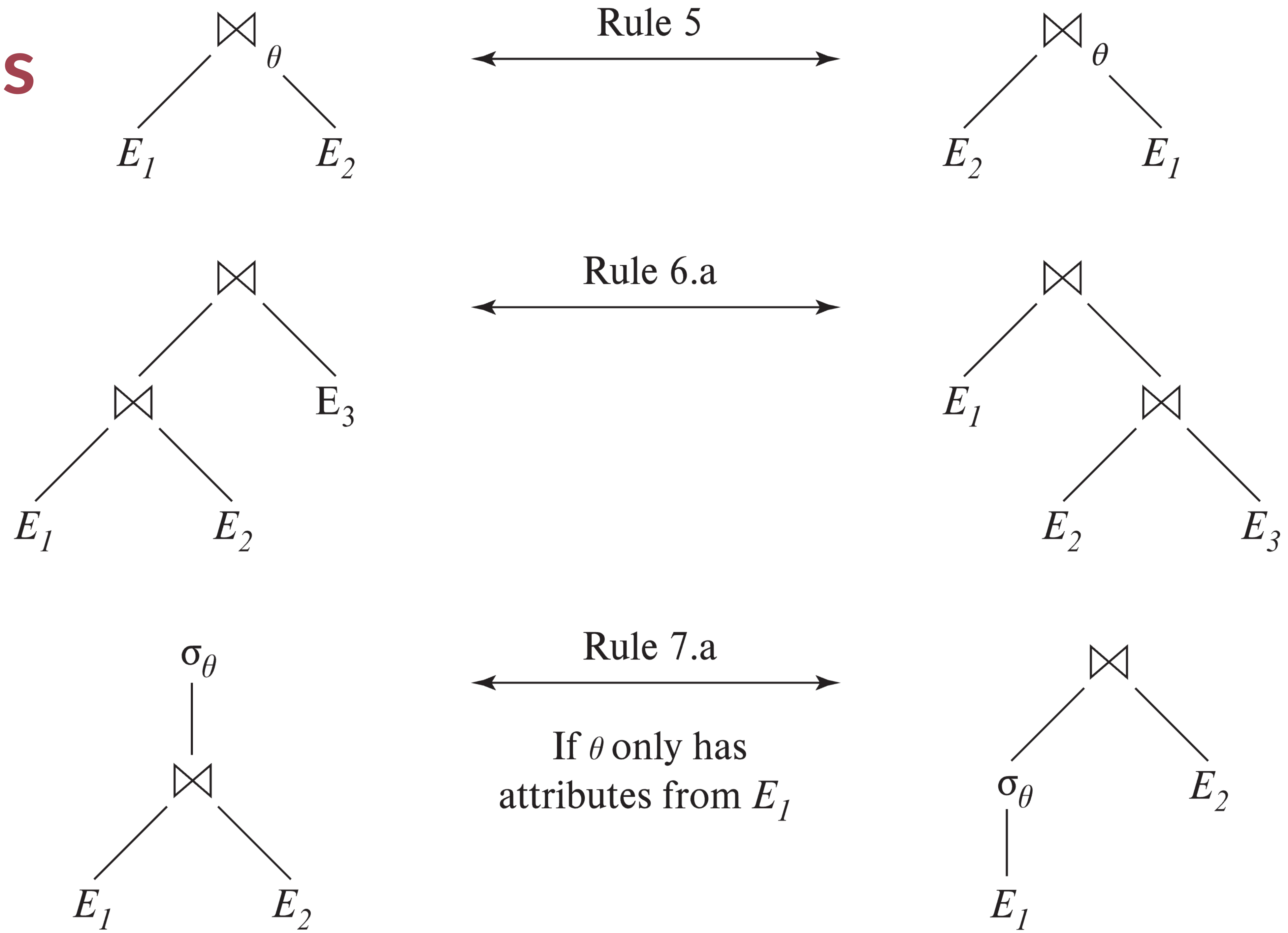
(b) Theta joins are associative in the following manner:

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 \equiv E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

where θ_2 involves attributes from only E_2 and E_3

Equivalence Rules

- Representation of Rule 5, 6(a) and 6(b) with diagrams



Equivalence Rules

7. The selection operation distributes over the theta join operation under the following two conditions:

- (a) When all the attributes in θ_0 involve only the attributes of one of the expressions (E_1) being joined:

$$\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) \equiv (\sigma_{\theta_0}(E_1)) \bowtie_\theta E_2$$

- (b) When θ_1 involves only the attributes of E_1 and θ_2 involves only the attributes of E_2:

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) \equiv (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2))$$

Equivalence Rules

8. The projection operation distributes over the theta join operation as follows:

(a) If θ involves only attributes from $L_1 \cup L_2$:

- Let L_1 and L_2 be sets of attributes from E_1 and E_2, respectively,

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

(b) In general, consider a join $E_1 \bowtie_\theta E_2$:

- Let L_1 and L_2 be sets of attributes from E_1 and E_2, respectively,
- Let L_3 be attributes of E_1 that are involved in join condition θ, but are not in L_1, and,
- Let L_4 be attributes of E_2 that are involved in join condition θ, but are not in L_2:

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

* Similar equivalences hold for left, right, and full outer join operations: ⟕, ⟖, and ⟗

Equivalence Rules

9. The set operations union and intersection are commutative

$$E_1 \cup E_2 \equiv E_2 \cup E_1$$
$$E_1 \cap E_2 \equiv E_2 \cap E_1$$

- However, set difference is not commutative

10. Set union and intersection are associative

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

Equivalence Rules

11. The selection operation distributes over $\cup$, $\cap$ and –

(a) $$\sigma_\theta (E_1 \cup E_2) \equiv \sigma_\theta (E_1) \cup \sigma_\theta(E_2)$$

(b) $$\sigma_\theta (E_1 \cap E_2) \equiv \sigma_\theta (E_1) \cap \sigma_\theta(E_2)$$

(c) $$\sigma_\theta (E_1 - E_2) \equiv \sigma_\theta (E_1) - \sigma_\theta(E_2)$$

(d) $$\sigma_\theta (E_1 \cap E_2) \equiv \sigma_\theta(E_1) \cap E_2$$

(e) $$\sigma_\theta (E_1 - E_2) \equiv \sigma_\theta(E_1) - E_2$$

12. The projection operation distributes over union

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

Transformation Example: Pushing Selections

- Query: Find the names of all instructors in the Music department, along with the titles of the courses (in the Music department) that they teach

```
select name, title
from instructor natural join (teaches natural join course
where dept_name = 'Music';
```

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

- Transformation using rule 7(a):
  - $\Pi_{name,\ title}((\sigma_{dept\_name=\ \text{'Music'}}\ (instructor)) \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course)))$

Perform selection <u>as early as possible</u> reduces the size of the relation to be joined

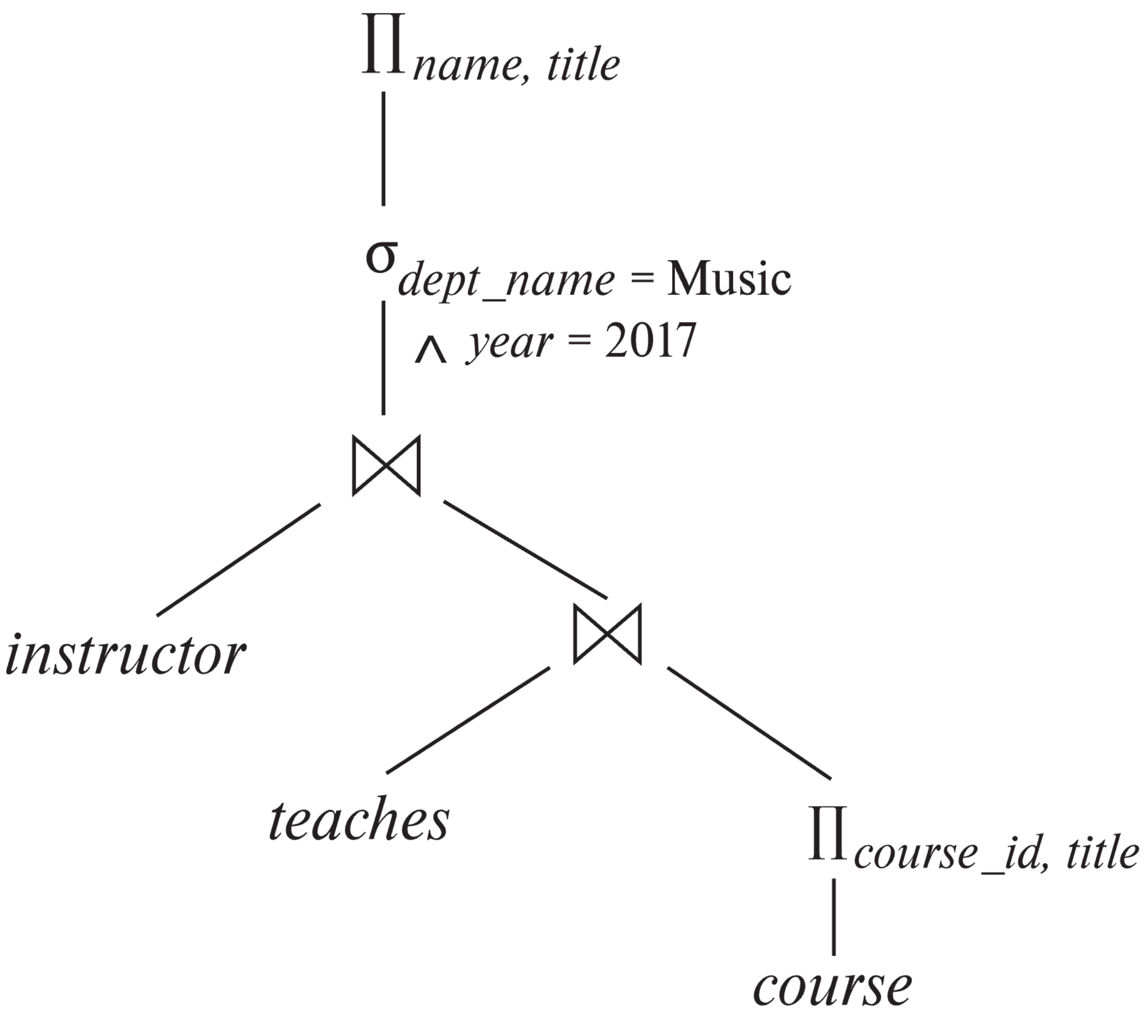
Transformation Example: Multiple Transformations

- Query: Find the names of all instructors in the Music department who have taught a course in 2017, along with the titles of the courses that they taught
  - $\Pi_{name,\ title}(\sigma_{dept\_name= \text{"Music"} \wedge year = 2017}\ (instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course))))$

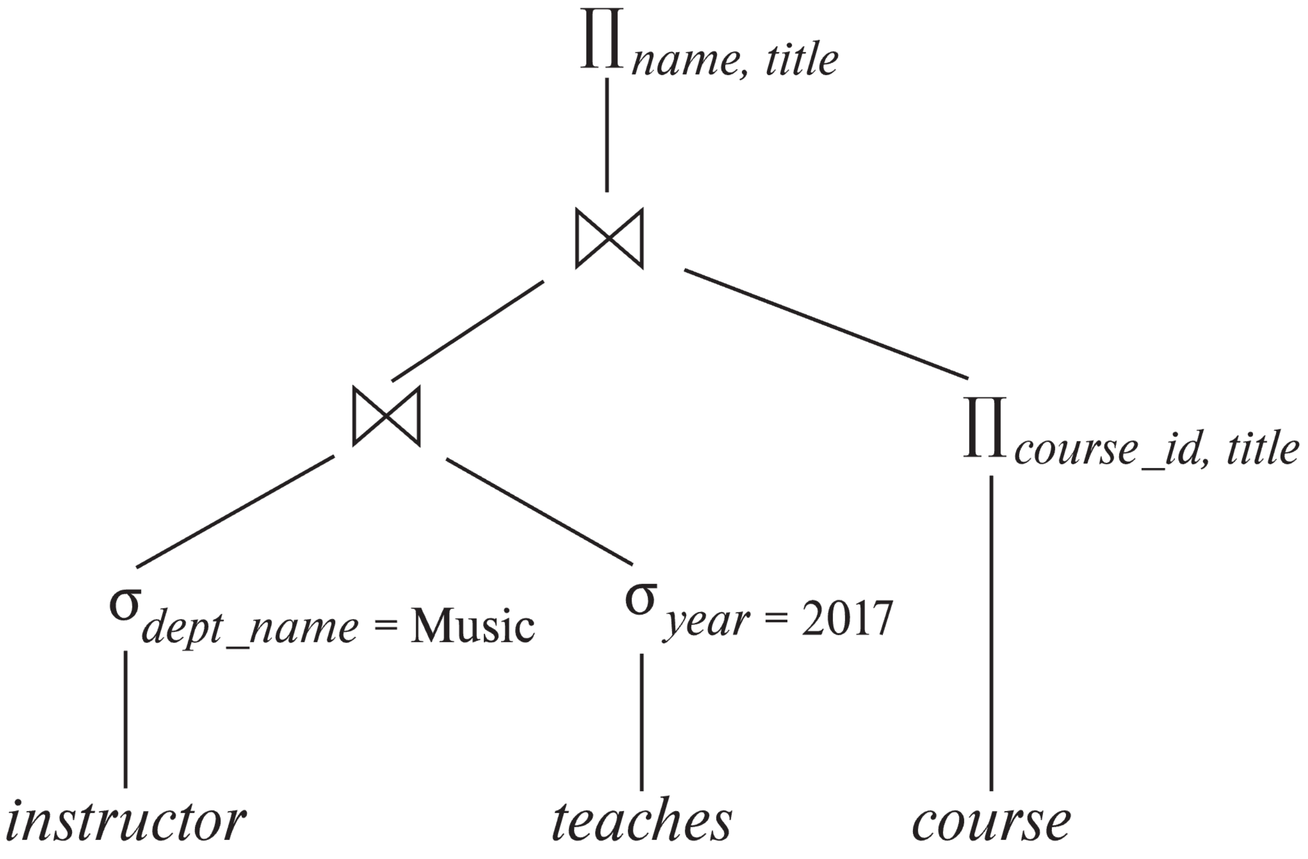
- Transformation using join associatively (Rule 6(a)):
  - $\Pi_{name,\ title}(\sigma_{dept\_name= \text{"Music"} \wedge year = 2017}\ ((instructor \bowtie teaches) \bowtie \Pi_{course\_id,\ title}\ (course)))$

- Second form provides an opportunity to apply the "perform selections early" rule, resulting in the subexpression:
  - $\sigma_{dept\_name\ =\ \text{"Music"}}\ (instructor) \bowtie \sigma_{\ year\ =\ 2017}\ (teaches)$

Transformation Example: Multiple Transformations



(a) Initial expression tree



(b) Tree after multiple transformations

* Transformation Example: Pushing Projections

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

- When we compute

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

  we obtain a relation whose schema is:

  (ID, name, dept_name, salary, course_id, sec_id, semester, year)

- Push projections using equivalence rules 8a and 8b; eliminate unneeded attributes from intermediate results to get:

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

Perform projections <u>as early as possible</u> reduces the size of the relation to be joined

Join Ordering Example

- For all relations r_1, r_2, and r_3,

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

  - * (Join Associativity) $\bowtie$

- If $r_2 \bowtie r_3$ is quite large and $r_1 \bowtie r_2$ is small, we choose

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

  so that we compute and store a smaller temporary relation

Cost Estimation

- Cost difference between evaluation plans for a query can be enormous
  - E.g., seconds vs. days in some cases

- Steps in cost-based query optimization
  - 1. Generate logically equivalent expressions using equivalence rules
  - 2. Annotate resultant expressions to get alternative query plans
  - 3. Choose the cheapest plan based on estimated cost

Cost Estimation

- Estimation of plan cost based on:
  - Statistical information about relations, such as:
    - number of tuples, number of distinct values for an attribute
  - Statistics estimation for intermediate results
    - to compute cost of complex expressions
  - Cost formulae for algorithms, computed using statistics

For more, please refer to Section 16.3 “Estimating Statistics of Expression Results” in the reference textbook