Query Processing and Query Optimization

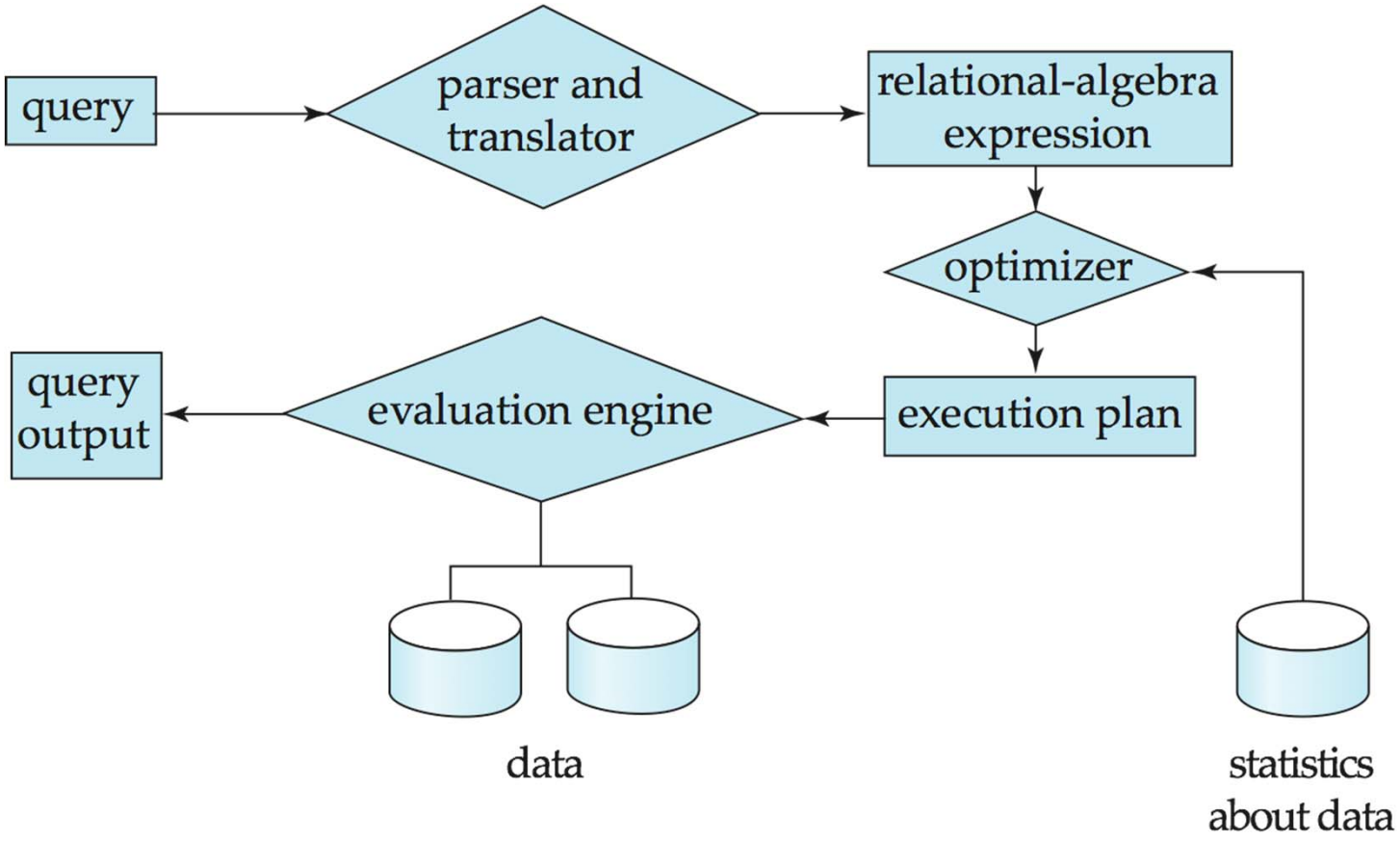
Database System Concepts, 6th Ed.

Query Processing

- **Query processing** refers to the range of activities involved in extracting data from a database. The activities include
  - Translation of queries in high-level database languages into expressions that can be used at the physical level of the file system
  - A variety of query-optimizing transformations
  - Actual evaluation of queries

Basic Steps in Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Basic Steps in Query Processing (Cont.)

- Parsing and translation
  - Before query processing can begin, the system must translate the query into a usable form
    - SQL is suitable for human use, but is ill suited to be the system's internal representation of a query
  - Translate the query into its internal form. This is then translated into relational algebra
  - Parser checks syntax, verifies relations

Basic Steps in Query Processing (Cont.)

- Optimization
  - Given a query, there are generally a variety of methods for computing the answer
  - Find the query-evaluation plan with the lowest cost
- Evaluation
  - The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query

Optimization

- A relational algebra expression may have many equivalent expressions
  - E.g., $\sigma_{salary<75000}(\Pi_{salary}(instructor))$ is equivalent to $\Pi_{salary}(\sigma_{salary<75000}(instructor))$
- Each relational algebra operation can be evaluated using one of several different algorithms
  - Correspondingly, a relational-algebra expression can be evaluated in many ways
- Annotated expression specifying detailed evaluation strategy is called an **evaluation-plan**
  - E.g., can use an index on *salary* to find instructors with salary < 75000,
  - Or, can perform complete relation scan and discard instructors with salary ≥ 75000

Optimization (Cont.)

- **Query Optimization**: Amongst all equivalent evaluation plans choose the one with lowest cost
  - Cost is estimated using statistical information from the database catalog
    - For example, number of tuples in each relation, size of tuples, etc.
- Related topics
  - How to measure query costs
  - Algorithms for evaluating relational algebra operations
  - How to combine algorithms for individual operations in order to evaluate a complete expression
  - We study how to optimize queries; that is, how to find an evaluation plan with lowest estimated cost

Measures of Query Cost

- Cost is generally measured as total elapsed time for answering query
  - Exact cost is hard to compute, rough estimate is possible
  - Many factors contribute to “time” cost
    - Disk *accesses,* CPU, Network communication
- Typically disk access is the predominant cost, and is also relatively easy to estimate. Measured by taking into account
  - Number of seeks * average-seek-cost
  - Number of blocks read * average-block-read-cost
  - Number of blocks written * average-block-write-cost
    - Cost to write a block is greater than cost to read a block (e.g., twice)
      - data is read back after being written to ensure that the write was successful

Measures of Query Cost (Cont.)

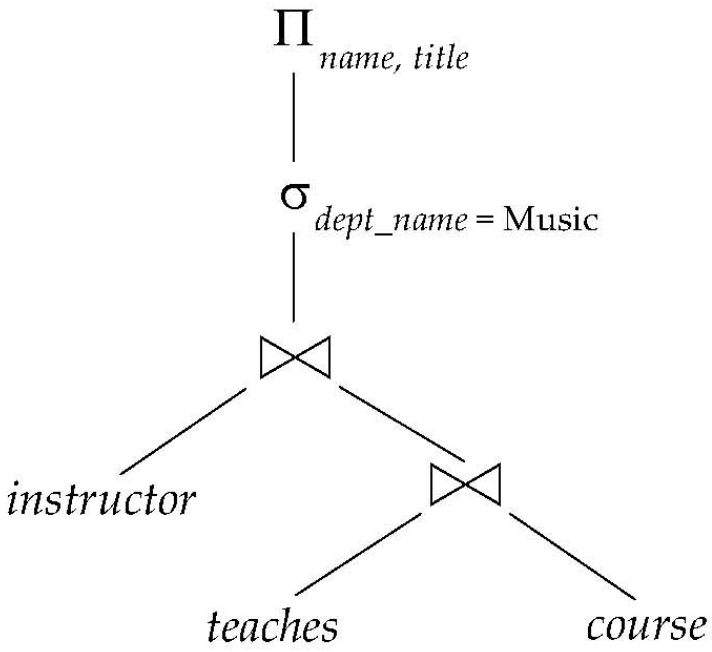
- The **response time** for a query-evaluation plan (that is, the wall-clock time required to execute the plan), assuming no other activity is going on in the computer, would account for all these costs, and could be used as a measure of the cost of the plan
- Very hard to estimate without actually executing the plan
  - Depends on the contents of the buffer when the query begins execution
    - Such information not available when optimizing queries
  - In a system with multiple disks, the response time depends on how accesses are distributed among disks
    - Hard to estimate without detailed knowledge of data layout on disk

Query Optimization

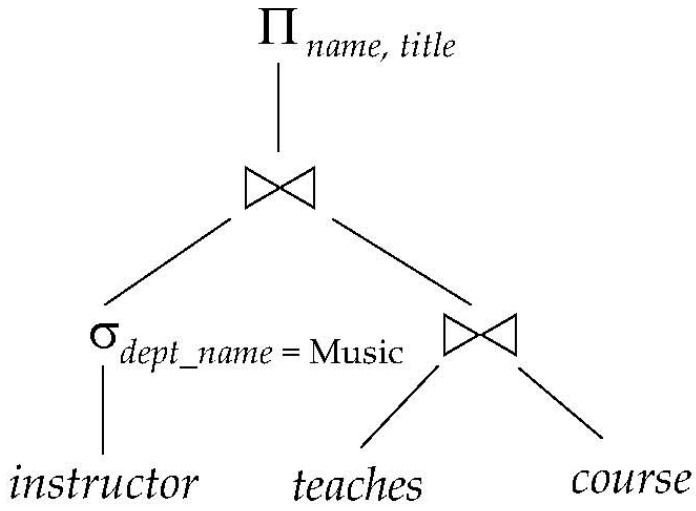
Database System Concepts, 6th Ed.

Equivalent Expressions

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



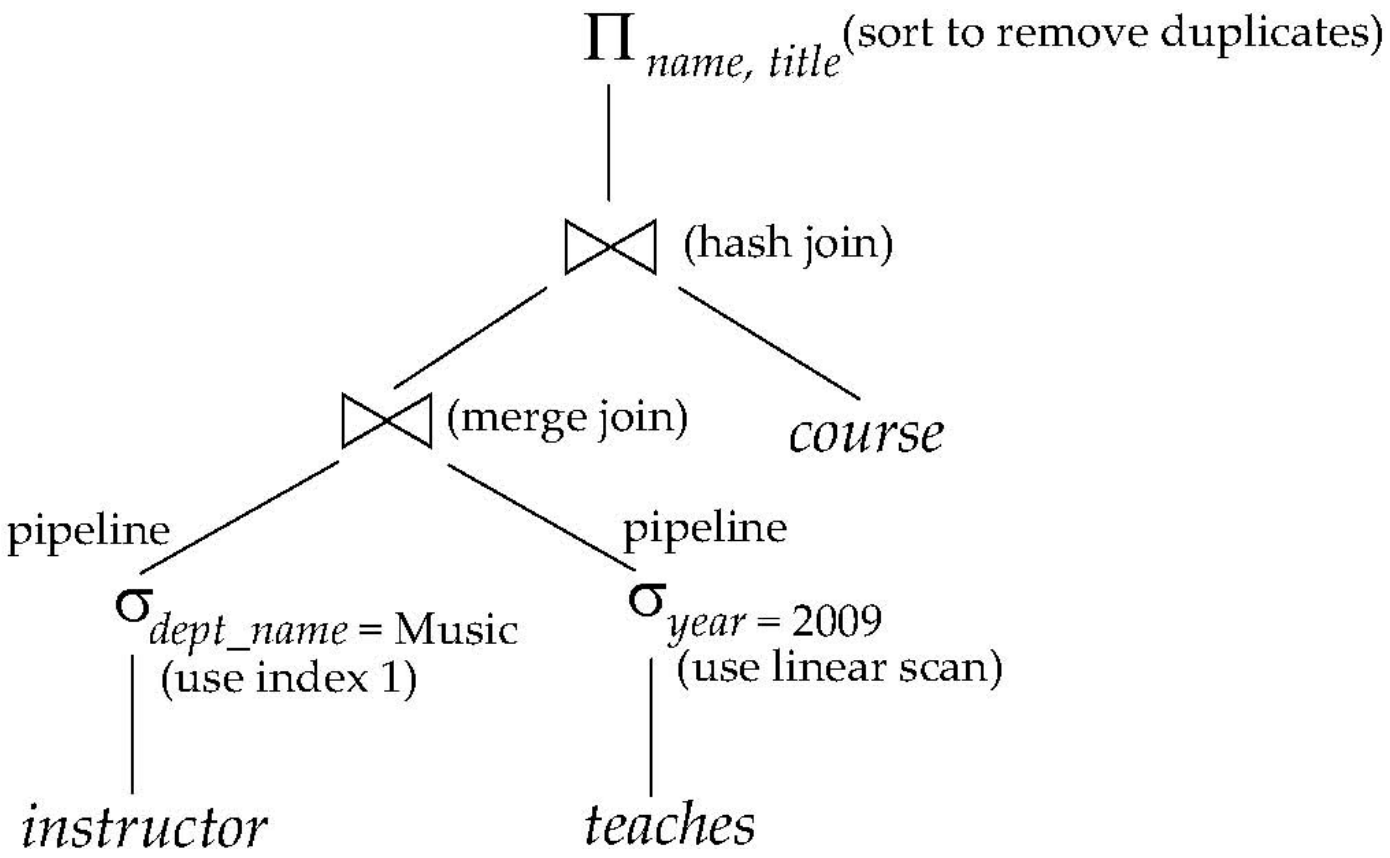
Initial expression tree



Transformed expression tree

Evaluation Plan

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



Cost Based Optimization

- Cost difference between evaluation plans for a query can be enormous
  - 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**
- Estimation of plan cost based on:
  - Statistical information about relations.
    - Number of tuples, number of distinct values for an attribute, etc.
  - Statistics estimation for intermediate results
    - To compute cost of complex expressions
  - Cost formulae for algorithms, computed using statistics

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
- An **equivalence rule** says that expressions of two forms are equivalent
  - Can replace expression of first form by second, or vice versa

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

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

   where $L_1 \subseteq L_2 \subseteq L_3 \ldots \subseteq L_n$

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

   1. $\sigma_{\theta}(E_1 \text{ X } E_2) = E_1 \bowtie_{\theta} E_2$ (definition of the theta join)
   2. $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

Equivalence Rules (Cont.)

5. Theta-join operations are commutative

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

6. Theta joins are associative in the following manner:

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

where θ_2 involves attributes from only E_2 and E_3

Equivalence Rules (Cont.)

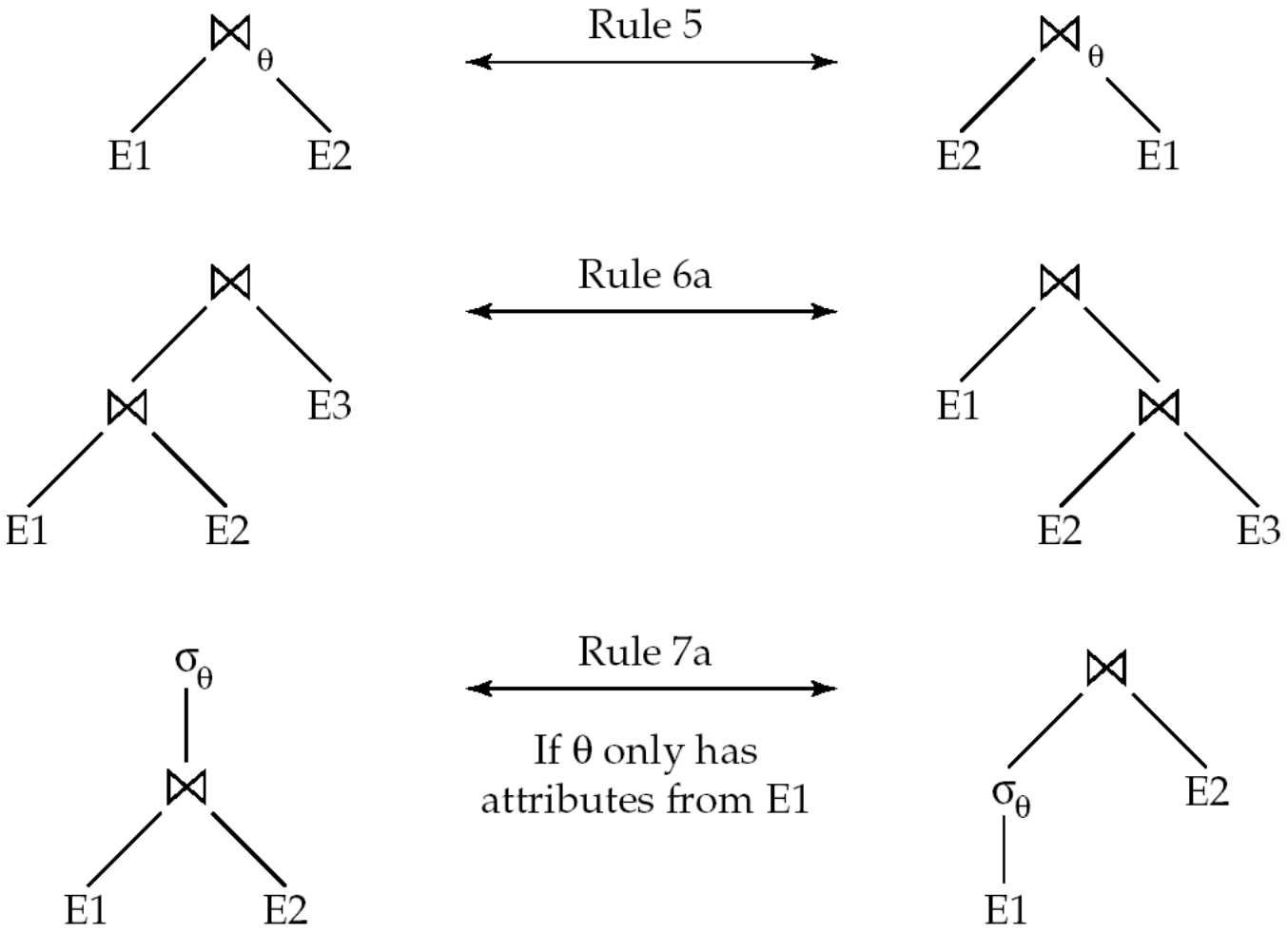
7. The selection operation distributes over the theta join operation under the following two conditions:
   (a) When all the attributes in θ_0 involve only the attributes of one of the expressions (E_1) being joined

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

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

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

Pictorial Depiction of Equivalence Rules



Transformation Example

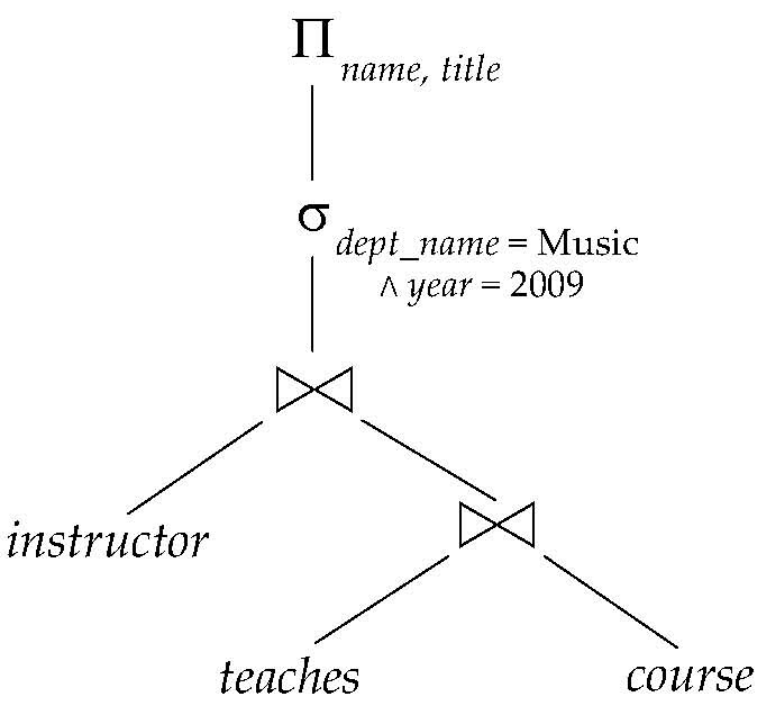
- Query: Find the names of all instructors in the Music department, along with the titles of the courses that they teach
  - $\Pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”}} (instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title} (course))))$
- Transformation using rule 7a.
  - $\Pi_{name,\ title}((\sigma_{dept\_name = \text{“Music”}}(instructor)) \bowtie (teaches \bowtie \Pi_{course\_id,\ title} (course)))$
- Above example suggests that
  - Performing the selection as early as possible reduces the size of the relation to be joined

Example with Multiple Transformations

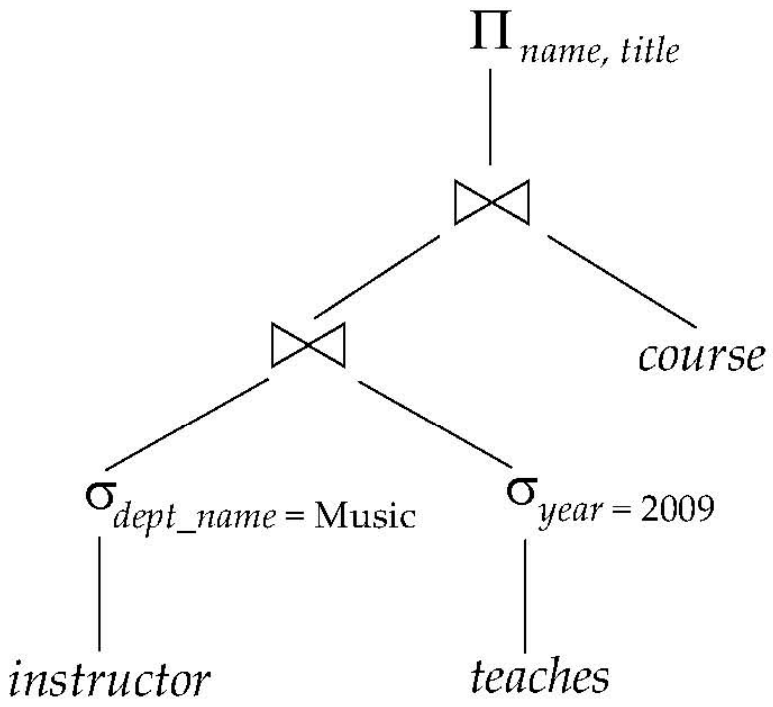
- Query: Find the names of all instructors in the Music department who have taught a course in 2009, along with the titles of the courses that they taught
  - $\Pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”} \wedge year = 2009}$
    $(instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course))))$
- Transformation using join associatively (Rule 6a):
  - $\Pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”} \wedge year = 2009}$
    $((instructor \bowtie teaches) \bowtie \Pi_{course\_id,\ title}\ (course)))$
- Second form provides an opportunity to apply the “perform selections early” rule, resulting in the subexpression

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

Multiple Transformations (Cont.)



(a) Initial expression tree



(b) Tree after multiple transformations

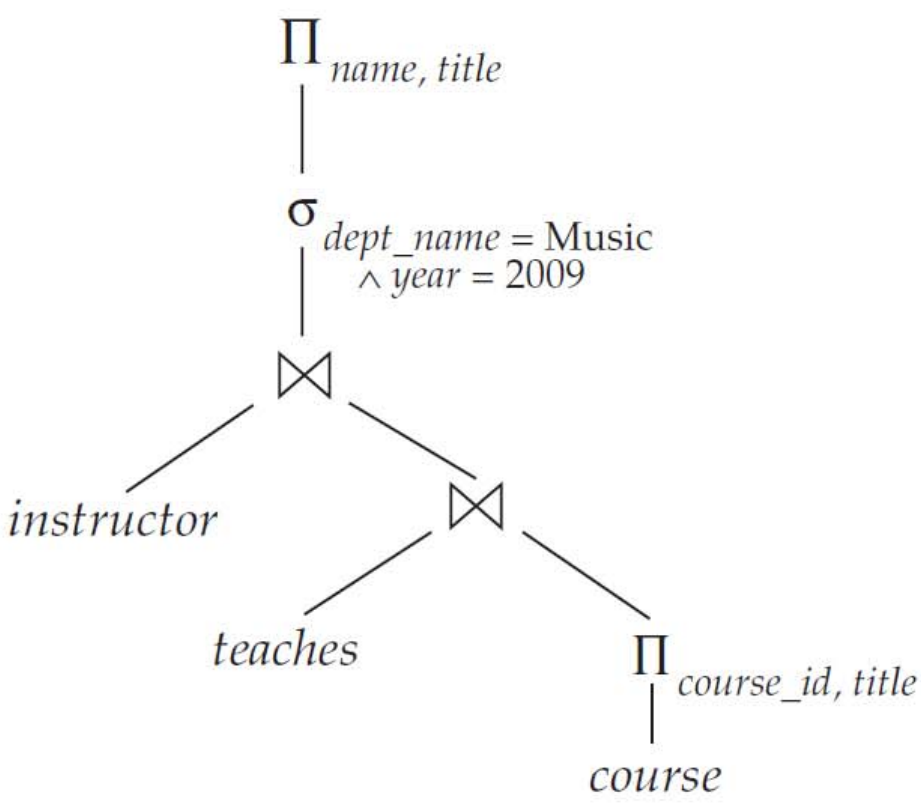
Transformation Example

- $\Pi_{name,\ title}((\sigma_{dept\_name=\text{“Music”}}(instructor) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course))$
- 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*)
- We can eliminate unneeded attributes from intermediate results to get:

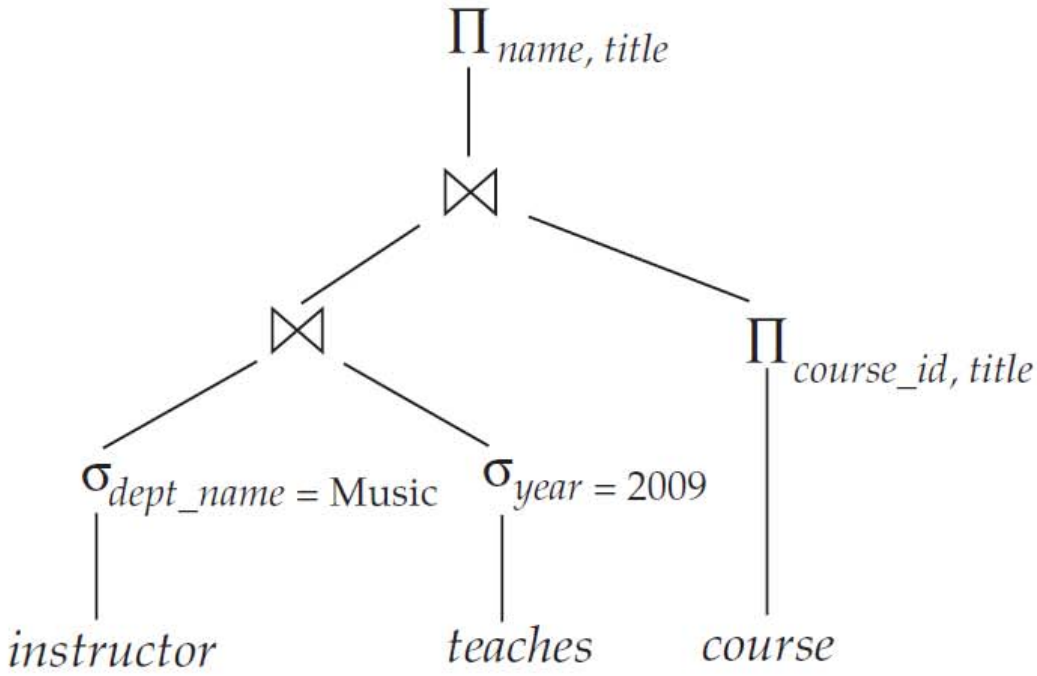
  $$\Pi_{name,\ title}((\Pi_{name,\ course\_id}((\sigma_{dept\_name=\text{“Music”}}(instructor)) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course))$$

- The projection $\Pi_{name,\ course\_id}$ reduces the size of the intermediate join results
- Above example suggests that:
  - Performing the projection as early as possible reduces the size of the relation to be joined

Transformations Example (Cont.)



(a) Initial expression tree



(b) Tree after multiple transformations

Figure 13.4 Multiple transformations.

Choice of Evaluation Plans

- Query optimizers use equivalence rules to **systematically** generate expressions equivalent to the given expression
- Can generate all equivalent expressions
  - very expensive in space and time
- Practical query optimizers incorporate elements of the following two broad approaches:
  - Search all the plans and choose the best plan in a cost-based fashion
  - Uses heuristics to choose a plan

Heuristic Optimization

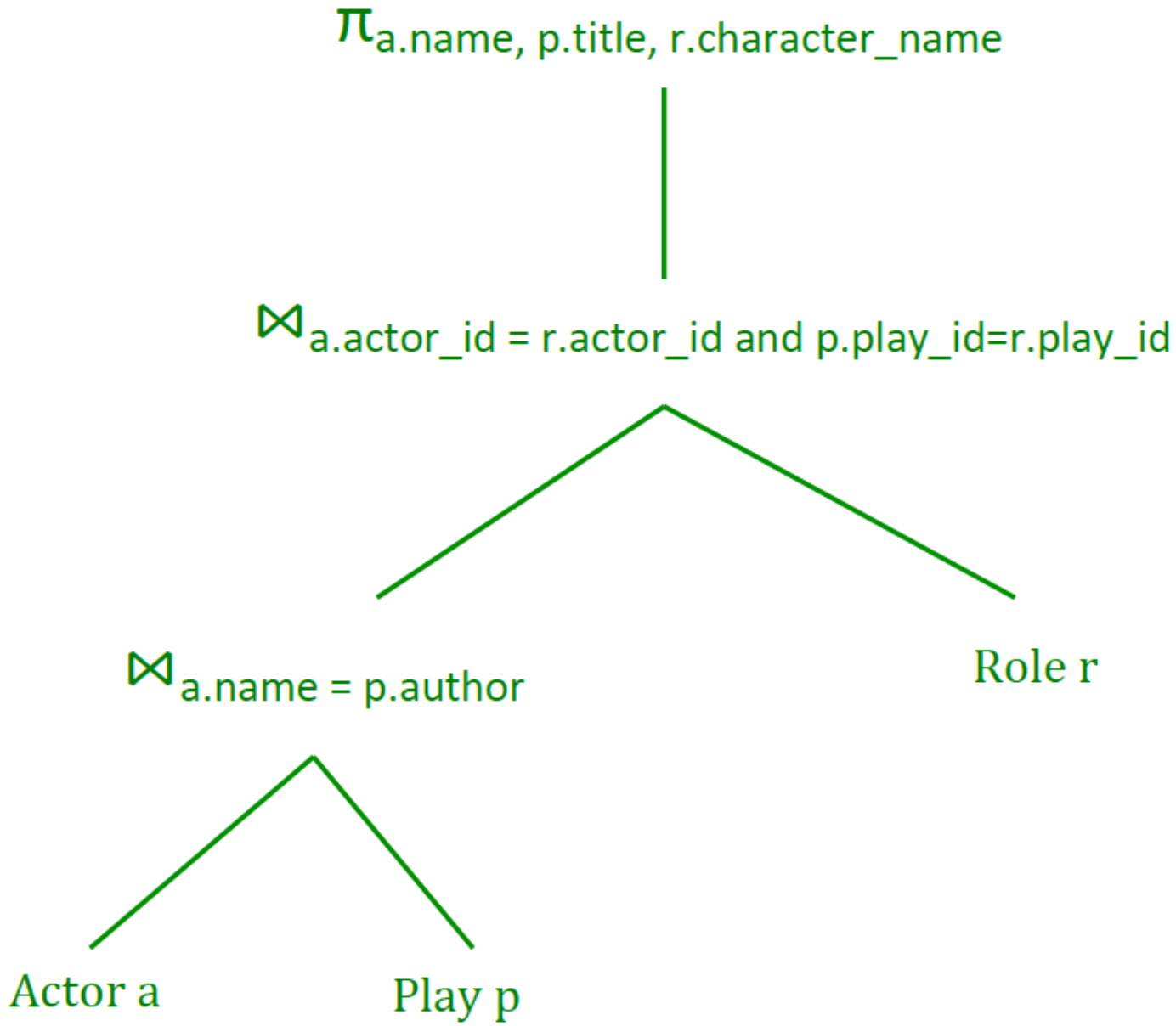
- Cost-based optimization is expensive
- Systems may use *heuristics* to reduce the number of choices that must be made in a cost-based fashion.
- Heuristic optimization transforms the query-tree by using a set of rules that typically (but not in all cases) improve execution performance:
  - Perform selection early (reduces the number of tuples)
  - Perform projection early (reduces the number of attributes)
  - Perform most restrictive selection and join operations (i.e., with smallest result size) before other similar operations
  - Some systems use only heuristics, others combine heuristics with partial cost-based optimization

Discussion

The following query returns information about all persons who have acted in a play that they have written. Draw an evaluation plan

```
SELECT a.name, p.title, r.character_name
FROM Actor a, Play p, Role r
Where a.name = p.author AND a.actor_id = r.actor_id AND p.play_id = r.play_id
```

Discussion



SQL Server Execution Plan Operations

- https://use-the-index-luke.com/sql/explain-plan/sql-server/operations