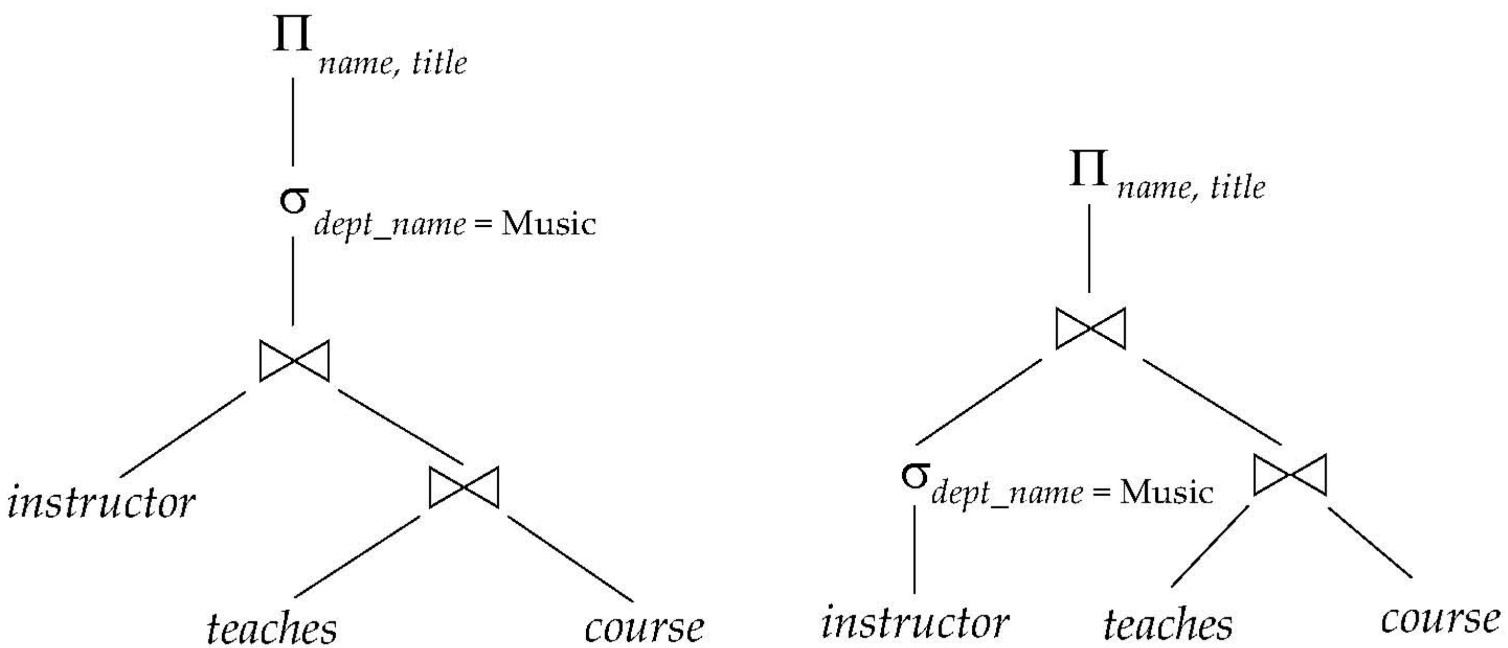
Chapter 13: Query Optimization

- Introduction
- Transformation of Relational Expressions
- Catalog Information for Cost Estimation
- Statistical Information for Cost Estimation
- Cost-based optimization
- Choosing Evaluation Plans

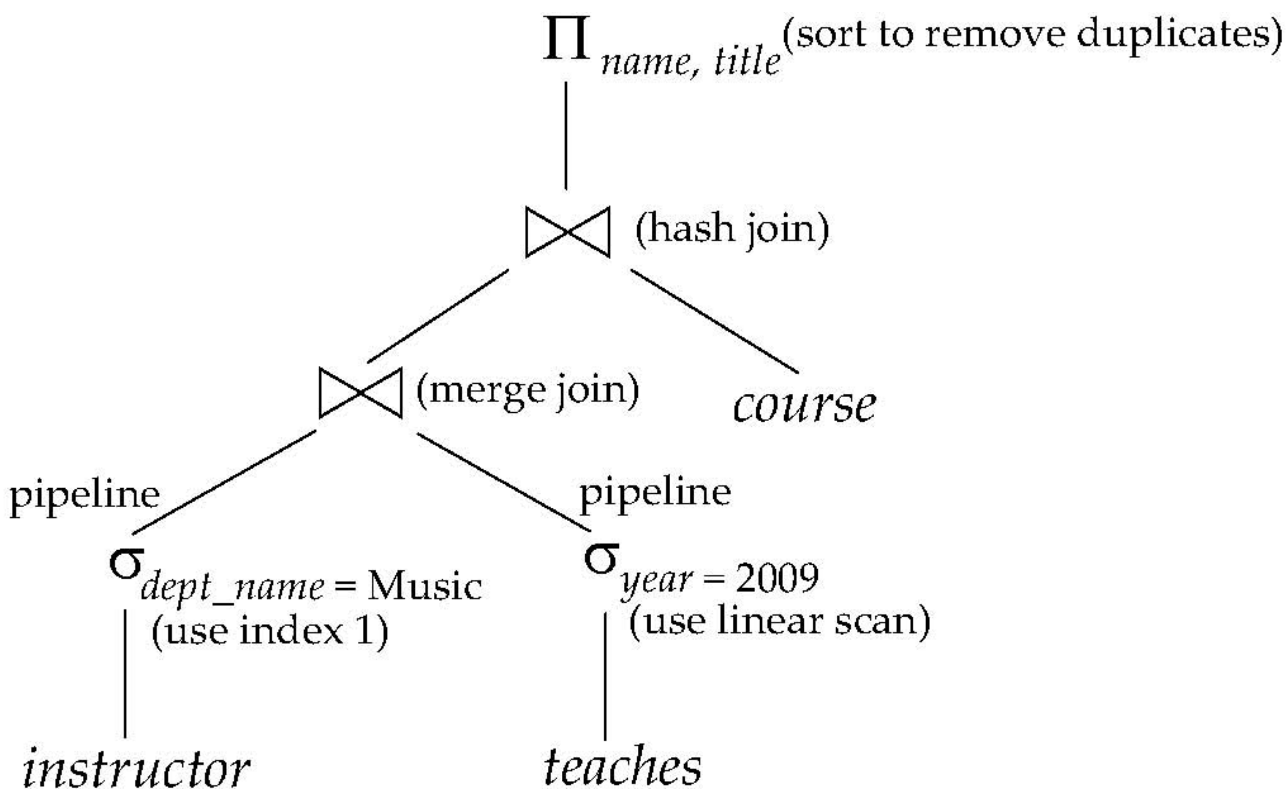
Introduction

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



Introduction (Cont.)

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



- Find out how to view query execution plans on your favorite database

Introduction (Cont.)

- 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**
- Estimation of plan cost based on:
  - Statistical information about relations. Examples:
    - number of tuples, number of distinct values for an attribute
  - Statistic 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
  - we don't care if they generate different results on databases that violate integrity constraints
- In SQL, inputs and outputs are multisets of tuples
  - Two expressions in the multiset version of the relational algebra are said to be equivalent if the two expressions generate the same multiset of tuples on every legal database instance.
- 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.

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

4. Selections can be combined with Cartesian products and theta joins.
   a. $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
   b. $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

Equivalence Rules (Cont.)

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

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

6. (a) Natural join operations are associative:

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

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

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

where θ_2 involves attributes from only E_2 and E_3.

Equivalence Rules (Cont.)

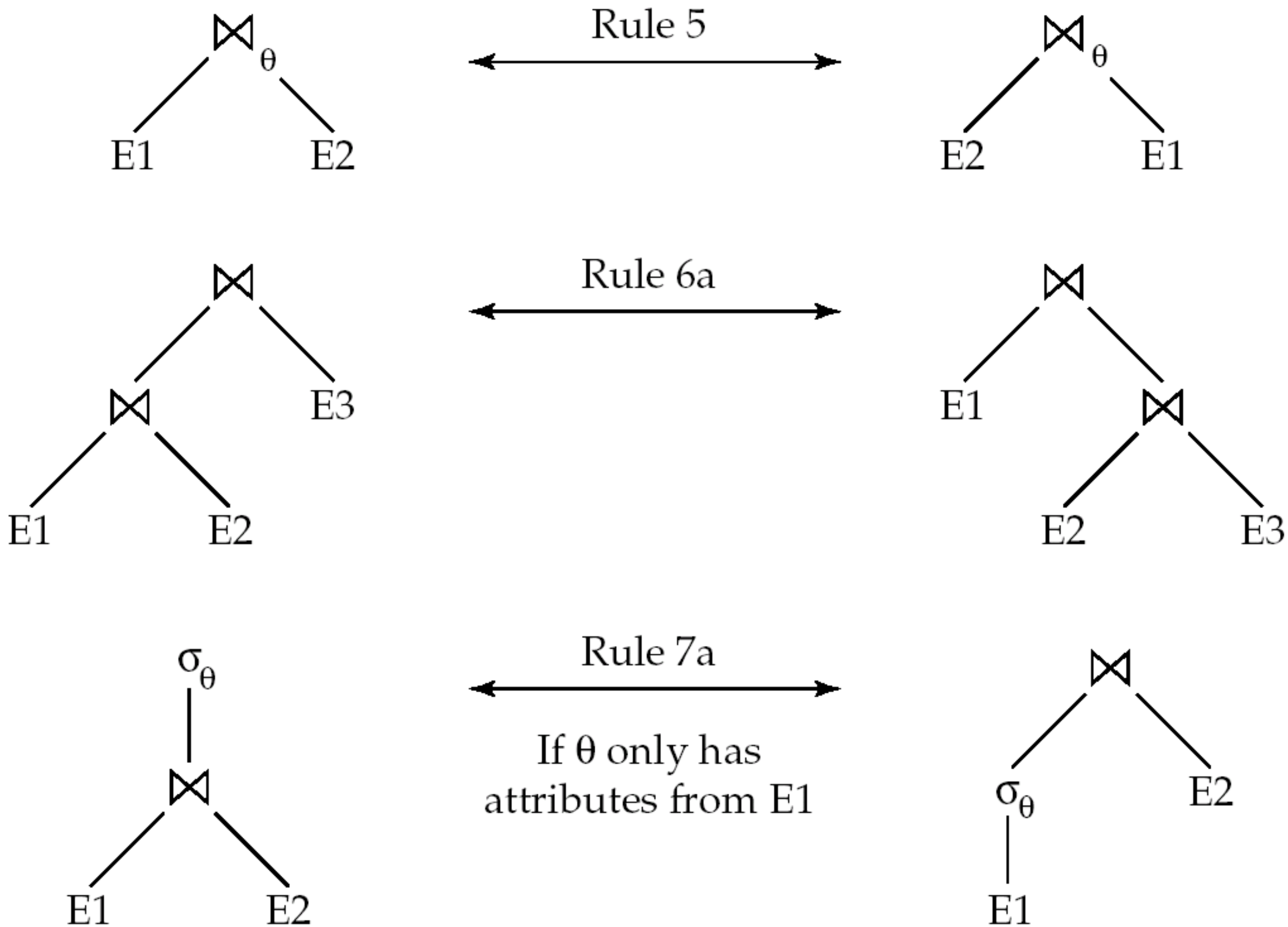
7. The selection operation distributes over the theta join operation under the following two conditions:
   (a) When all the attributes in θ_0 involve only the attributes of one of the expressions (E_1) being joined.

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

   (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



Equivalence Rules (Cont.)

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

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

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

   (b) 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 $L_1 \cup L_2$, and
   - let L_4 be attributes of E_2 that are involved in join condition θ, but are not in $L_1 \cup L_2$.

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

Equivalence Rules (Cont.)

9. The set operations union and intersection are commutative

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

- (set difference is not commutative).

10. Set union and intersection are associative.

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

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

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

and similarly for $\cup$ and $\cap$ in place of $-$

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

and similarly for $\cap$ in place of $-$, but not for $\cup$

12. The projection operation distributes over union

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

Transformation Example: Pushing Selections

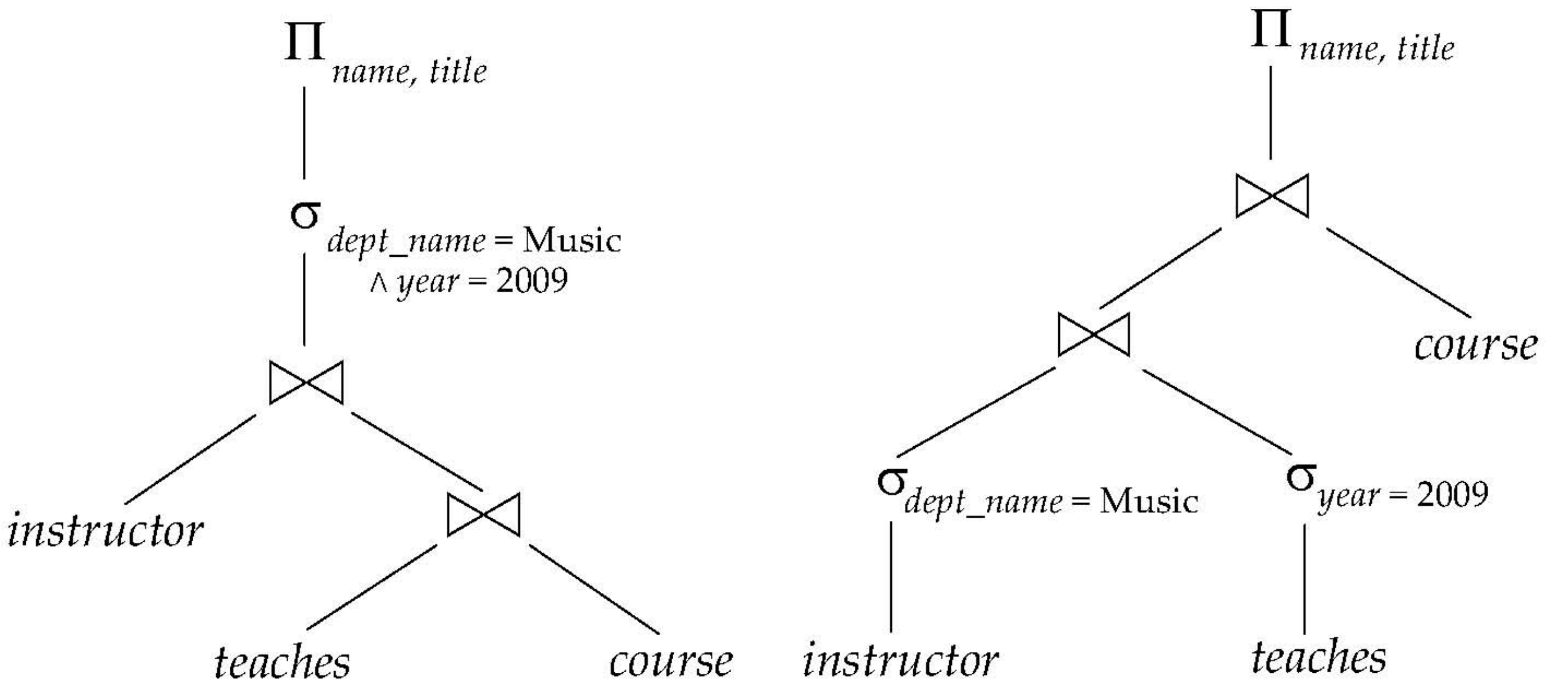
- Query: Find the names of all instructors in the Music department, along with the titles of the courses that they teach
  - $\Pi_{name,\ title}(\sigma_{dept\_name=\text{“Music”}}$
    $(instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}(course))))$
- Transformation using rule 7a.
  - $\Pi_{name,\ title}((\sigma_{dept\_name=\text{“Music”}}(instructor)) \bowtie$
    $(teaches \bowtie \Pi_{course\_id,\ title}(course)))$
- 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):
  - $\Pi_{name,\ title}(\sigma_{dept\_name=\text{"Music"} \wedge gear = 2009}$ $((instructor \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course)))$
- 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: 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:

  $$\Pi_{name,\ title}(\Pi_{name,\ course\_id}(\sigma_{dept\_name=\ \text{“Music”}}(instructor) \bowtie teaches)) \bowtie \Pi_{course\_id,\ title}(course))$$
- Performing the projection as early as possible 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)
- 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.

Join Ordering Example (Cont.)

- Consider the expression

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

- Could compute $teaches \bowtie \Pi_{course\_id,\ title}(course)$ first, and join result with

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

but the result of the first join is likely to be a large relation.

- Only a small fraction of the university’s instructors are likely to be from the Music department
  - it is better to compute

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

first.

Enumeration of Equivalent Expressions

- Query optimizers use equivalence rules to **systematically** generate expressions equivalent to the given expression
- Can generate all equivalent expressions as follows:
  - Repeat
    - apply all applicable equivalence rules on every subexpression of every equivalent expression found so far
    - add newly generated expressions to the set of equivalent expressions

    Until no new equivalent expressions are generated above
- The above approach is very expensive in space and time
  - Two approaches
    - Optimized plan generation based on transformation rules
      - Space requirements reduced by sharing common sub-expressions
      - Time requirements are reduced by not generating all expressions
    - Special case approach for queries with only selections, projections and joins

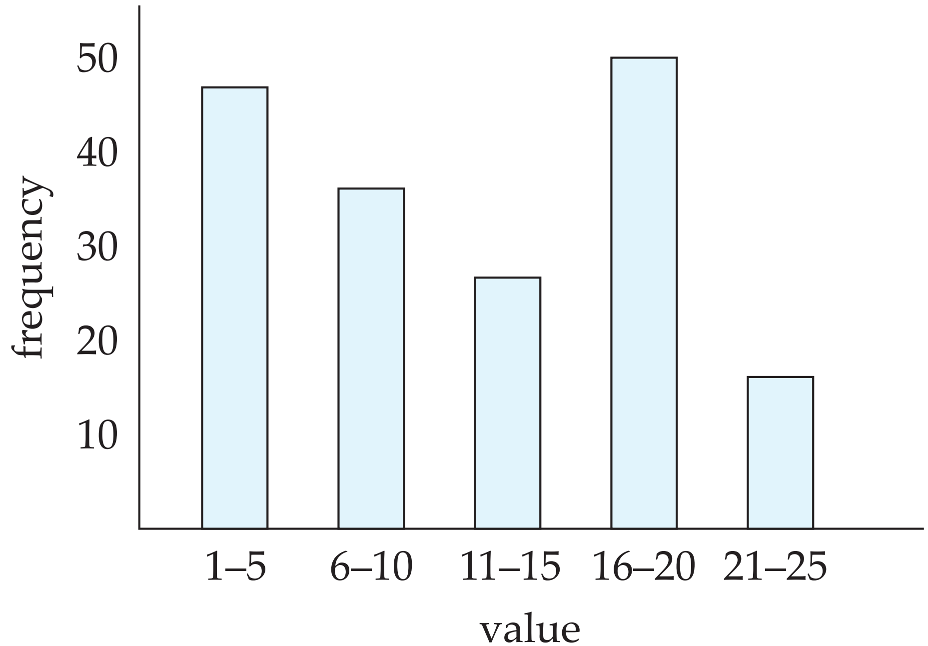
Statistical Information for Cost Estimation

- n_r: number of tuples in a relation *r.*
- b_r: number of blocks containing tuples of *r.*
- l_r: size of a tuple of *r.*
- f_r: blocking factor of *r* — i.e., the number of tuples of *r* that fit into one block.
- $V(A, r)$: number of distinct values that appear in *r* for attribute *A;* same as the size of $\prod_A(r)$.
- If tuples of *r* are stored together physically in a file, then:

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

Histograms

- Histogram on attribute *age* of relation *person*



- **Equi-width** histograms
- **Equi-depth** histograms

Selection Size Estimation

- $\sigma_{A=v}(r)$
  - $n_r / V(A,r)$: number of records that will satisfy the selection
  - Equality condition on a key attribute: *size estimate* = 1
- $\sigma_{A \le V}(r)$ (case of $\sigma_{A \geq V}(r)$ is symmetric)
  - Let c denote the estimated number of tuples satisfying the condition.
  - If min(A,r) and max(A,r) are available in catalog
    - c = 0 if v < min(A,r)
    - c = $n_r \cdot \dfrac{v - \min(A,r)}{\max(A,r) - \min(A,r)}$
  - If histograms available, can refine above estimate
  - In absence of statistical information *c* is assumed to be $n_r / 2$.

Size Estimation of Complex Selections

- The **selectivity** of a condition θ_i is the probability that a tuple in the relation *r* satisfies θ_i .
  - If s_i is the number of satisfying tuples in *r*, the selectivity of θ_i is given by s_i / n_r.
- **Conjunction:** $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n}(r)$. *Assuming indepdence,* estimate of tuples in the result is: $$n_r * \frac{s_1 * s_2 * \ldots * s_n}{n_r^n}$$
- **Disjunction:** $\sigma_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n}(r)$. Estimated number of tuples: $$n_r * \left(1 - (1 - \frac{s_1}{n_r}) * (1 - \frac{s_2}{n_r}) * \ldots * (1 - \frac{s_n}{n_r})\right)$$
- **Negation:** $\sigma_{\neg\theta}(r)$. Estimated number of tuples: $$n_r - size(\sigma_\theta(r))$$

Estimation of the Size of Joins

- The Cartesian product $r \times s$ contains $n_r . n_s$ tuples; each tuple occupies $s_r + s_s$ bytes.
- If $R \cap S = \emptyset$, then $r \bowtie s$ is the same as $r \times s$.
- If $R \cap S$ is a key for R, then a tuple of s will join with at most one tuple from r
  - therefore, the number of tuples in $r \bowtie s$ is no greater than the number of tuples in s.
- If $R \cap S$ *in* S is a foreign key in S referencing R, then the number of tuples in $r \bowtie s$ is exactly the same as the number of tuples in s.
    - The case for $R \cap S$ being a foreign key referencing S is symmetric.

Estimation of the Size of Joins (Cont.)

- If $R \cap S = \{A\}$ is not a key for R or S.
  If we assume that every tuple t in R produces tuples in $R \bowtie S$, the number of tuples in $R \bowtie S$ is estimated to be:

  $$\frac{n_r * n_s}{V(A,s)}$$

  If the reverse is true, the estimate obtained will be:

  $$\frac{n_r * n_s}{V(A,r)}$$

  The lower of these two estimates is probably the more accurate one.
- Can improve on above if histograms are available
  - Use formula similar to above, for each cell of histograms on the two relations

Size Estimation for Other Operations

- Projection: estimated size of $\prod_A(r) \; = \; V(A,r)$
- Aggregation : estimated size of ${}_A g_F(r) \; = V(A,r)$
- Set operations
  - For unions/intersections of selections on the same relation: rewrite and use size estimate for selections
    - E.g. $\sigma_{\theta 1}(r) \cup \sigma_{\theta 2}(r)$ can be rewritten as $\sigma_{\theta 1 \vee \theta 2}(r)$
  - For operations on different relations:
    - estimated size of $r \cup s$ = size of r + size of s.
    - estimated size of $r \cap s$ = minimum size of r and size of s.
    - estimated size of $r - s$ = r.
    - <u>All the three estimates may be quite inaccurate, but provide upper bounds on the sizes</u>.
- Outer join:
  - Estimated size of $r ⟕ s$ = *size of* $r \bowtie s$ + *size of* r
    - Case of right outer join is symmetric
  - Estimated size of $r ⟗ s$ = *size of* $r \bowtie s$ + size of r + size of s

Estimation of Number of Distinct Values

Selections: $\sigma_\theta (r)$

- If θ forces *A* to take a specified value: $V(A,\sigma_\theta (r)) = 1$.
  - e.g., $A = 3$
- If θ forces A to take on one of a specified set of values:
  $V(A,\sigma_\theta (r))$ = number of specified values.
  - (e.g., ($A = 1 \ V A = 3 \ V A = 4$)),
- If the selection condition θ is of the form *A op v*
  estimated $V(A,\sigma_\theta (r)) = V(A,r) * s$
  - where *s* is the selectivity of the selection.
- In all the other cases: use approximate estimate of
  $\min(V(A,r), n_{\sigma\theta (r)})$
  - More accurate estimate can be got using probability theory, but this one works fine generally

Estimation of Distinct Values (Cont.)

Joins: $r \bowtie s$

- If all attributes in A are from r
  estimated $V(A, r \bowtie s) = \min(V(A,r), n_{r \bowtie s})$
- If A contains attributes $A1$ from r and $A2$ from s, then estimated $V(A,r \bowtie s) =$

  $$\min(V(A1,r)^* V(A2 - A1,s),\ V(A1 - A2,r)^* V(A2,s),\ n_{r \bowtie s})$$

  - More accurate estimate can be got using probability theory, but this one works fine generally

Estimation of Distinct Values (Cont.)

- Estimation of distinct values are straightforward for projections.
  - They are the same in $\prod_{A}(r)$ as in *r*.
- The same holds for grouping attributes of aggregation.
- For aggregated values
  - For min(*A*) and max(*A*), the number of distinct values can be estimated as min(V(*A,r*), *V*(*G,r*)) where G denotes grouping attributes
  - For other aggregates, assume all values are distinct, and use *V*(*G,r*)

Cost Estimation

- Cost of each operator computed as described earlier
  - Need statistics of input relations
    - E.g. number of tuples, sizes of tuples
- Inputs can be results of sub-expressions
  - Need to estimate statistics of expression results
  - To do so, we require additional statistics
    - E.g. number of distinct values for an attribute

Choice of Evaluation Plans

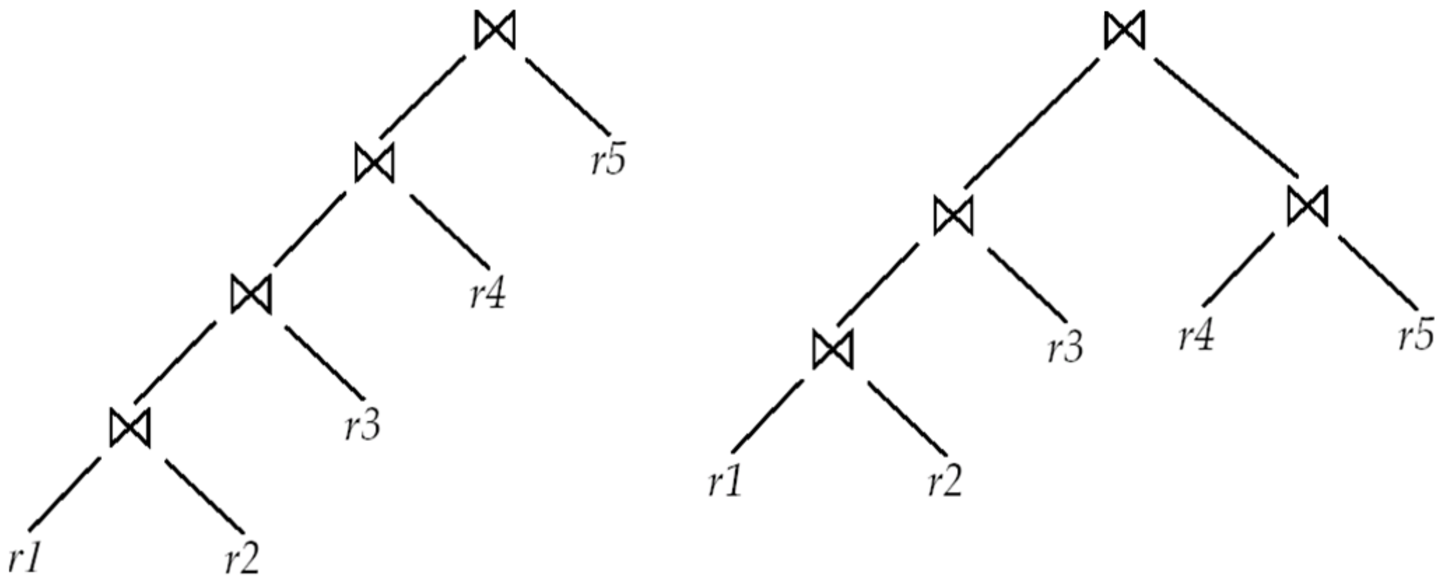
- Must consider the interaction of evaluation techniques when choosing evaluation plans
  - choosing the cheapest algorithm for each operation independently may not yield best overall algorithm. E.g.
    - merge-join may be costlier than hash-join, but may provide a sorted output which reduces the cost for an outer level aggregation.
    - nested-loop join may provide opportunity for pipelining
- Practical query optimizers incorporate elements of the following two broad approaches:
  1. Search all the plans and choose the best plan in a cost-based fashion.
  2. Uses heuristics to choose a plan.

Cost-Based Optimization

- Consider finding the best join-order for $r_1 \bowtie r_2 \bowtie \ldots r_n$.
- There are $(2(n - 1))!/(n - 1)!$ different join orders for above expression. With $n = 7$, the number is 665280, with $n = 10$, the number is greater than 176 billion!
- No need to generate all the join orders. Using dynamic programming, the least-cost join order for any subset of $\{r_1, r_2, \ldots r_n\}$ is computed only once and stored for future use.

Left Deep Join Trees

- In **left-deep join trees,** the right-hand-side input for each join is a relation, not the result of an intermediate join.



(a) Left-deep join tree (b) Non-left-deep join tree

Cost of Optimization

- With dynamic programming time complexity of optimization with bushy trees is $O(3^n)$.
  - With $n = 10$, this number is 59000 instead of 176 billion!
  - Space complexity is $O(2^n)$
- If only left-deep trees are considered, time complexity of finding best join order with dynamic programming is $O(n\ 2^n)$
  - Space complexity remains at $O(2^n)$
- Cost-based optimization is expensive, but worthwhile for queries on large datasets (typical queries have small n, generally < 10)

Heuristic Optimization

- Cost-based optimization is expensive, even with dynamic programming.
- 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.

Structure of Query Optimizers

- Many optimizers considers only left-deep join orders.
  - Plus heuristics to push selections and projections down the query tree
  - Reduces optimization complexity and generates plans amenable to pipelined evaluation.
- Heuristic optimization used in some versions of Oracle:
  - Repeatedly pick “best” relation to join next
    - Starting from each of n starting points. Pick best among these