Databases and Information Systems CS303

Query Optimization
27-10-2023

Query Optimization

- Find the names of all instructors in the Music department together with the course title of all the courses that the instructors teach

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

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Course

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

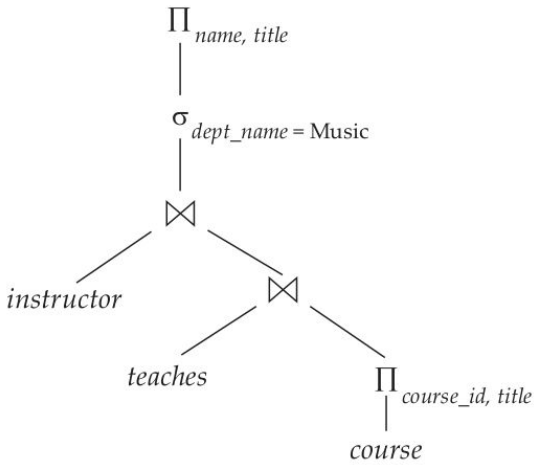
Teaches

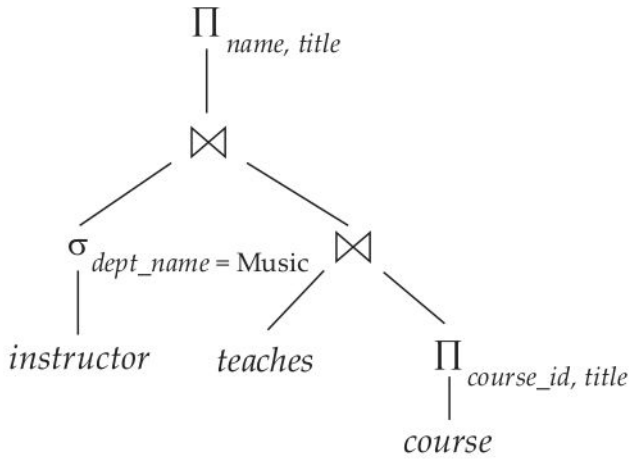
Query Optimization

- Find the names of all instructors in the Music department together with the course title of all the courses that the instructors teach

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

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





Query Optimization

- Given a relational-algebra expression, it is the job of the query optimizer to come up with:
  - Query-evaluation plan that computes the same result as the given expression
  - Least-costly way of generating the result

- Generation of query-evaluation plans involves three steps:

  - Generating expressions that are logically equivalent to the given expression

  - Annotating the resultant expressions in alternative ways to generate alternative query-evaluation plans

  - Estimating the cost of each evaluation plan

- In POSTGRES, EXPLAIN query which show query execution plan

Transformation of Relational Expressions

- Two relational-algebra expressions are said to be equivalent if, on every legal database instance, the two expressions generate the same set of tuples

- The order in which the tuples are generated are not relevant

- We will consider standard evaluation of relational algebraic expressions
  - Which eliminates duplicates

  - Exercise to extend it to multiset version

Transformation of Relational Expressions : Equivalence Rules

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

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

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

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

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

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

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

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

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

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

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

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

Transformation of Relational Expressions : Equivalence Rules

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

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

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

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

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2)$$

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - E_2$$

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

Transformation of Relational Expressions : Equivalence Rules

- Example :

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

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

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

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

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

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

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

Transformation of Relational Expressions : Equivalence Rules

- Example :

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

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

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

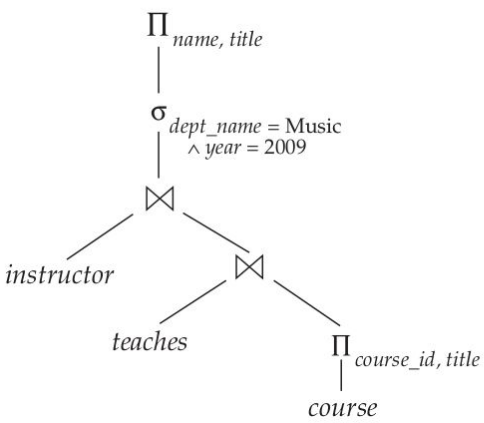
$$\sigma_{dept\_name = \text{“Music”}}\ (instructor) \bowtie \sigma_{year\ = 2009}\ (teaches)$$

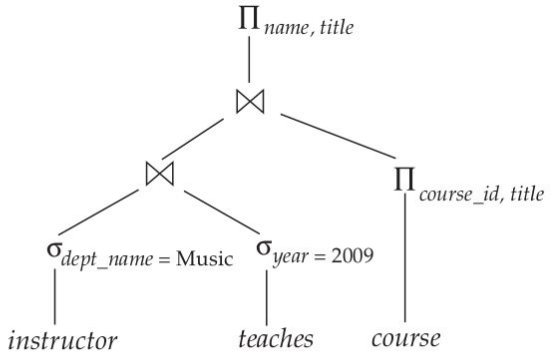
Transformation of Relational Expressions : Equivalence Rules

- Example :

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

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





Transformation of Relational Expressions : Equivalence Rules

- Goal is to reduce the size of the intermediate results
- We only get equivalent expressions
  - We do not know about the cost of these expressions
- Final selection determined by the Query Planner
  - Depending on various evaluation algorithms

Join Ordering

- Good ordering of join operations reduces the size of temporary results

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

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

$$teaches \bowtie \Pi_{course\_id, title}(course)$$ Generates lot of tuples

$$\sigma_{dept\_name = \text{“Music”}} (instructor) \bowtie teaches$$ Generates lesser number of tuples

Enumeration of Equivalent expressions

```
procedure genAllEquivalent(E)
EQ = {E}
repeat
    Match each expression E_i in EQ with each equivalence rule R_j
    if any subexpression e_i of E_i matches one side of R_j
        Create a new expression E' which is identical to E_i, except that
            e_i is transformed to match the other side of R_j
        Add E' to EQ if it is not already present in EQ
until no new expression can be added to EQ
```

- Above algorithm is costly in both space and time
- There are some ways to optimally generate the candidates for evaluation (later)

Estimating the statistics of Expression Results

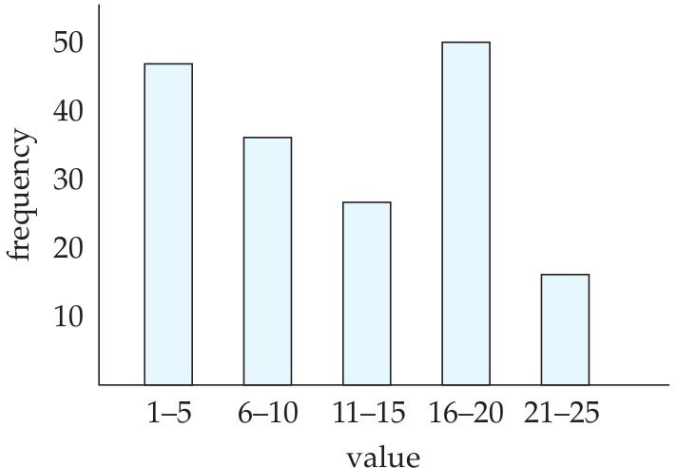
- Cost of an operation depends on the size and other statistics of its inputs
- Some statistics about database relations that are stored in database-system catalogs
- A query-evaluation plan that has the lowest estimated execution cost may therefore not actually have the lowest actual execution cost
  - But real-world experience has shown that estimates are either the lowest actual execution costs, or are close to the lowest actual execution costs.

Catalog Information

- Number of tuples in a relation
- Number of blocks containing the tuples of the relation
- Size of the tuples in the relation (bytes)
- Number of tuples that fit into one block
- Number of distinct values that appear in the relation r for attribute A
- Height of B^+ trees, number of leaf nodes for each of the index maintained

Catalog Information

- Updating catalog when database is modified is costly
- Does not happen on every modification
  - System updates catalog periodically
- Statistics are not fully accurate during query plan cost evaluation
  - But sufficiently accurate to provide a good estimation of the relative costs of the different plans
- Most database also store attribute count as histogram
  - Many other details too



Selection size estimation

- $\sigma_{A = a}(r)$
  - Estimate of the number of tuples = n_r / V(A,r)
    - n_r is the number of tuples in r
    - V(A,r) is the number of distinct values of A that occurs in r
  - This assumes all values of A occur with uniform frequency
    - Good approximate in real life
  - If histogram data is available then we can get a better estimate

Selection size estimation

- $\sigma_{A <= a}(r)$
  - Catalog maintains min(A,r) and max(A,r) be the lowest and highest values that A can take
  - Estimate of the number of tuples
    - If a < min(A,r) then estimate = 0
    - If a >= max(A,r) then estimate = n_r

$$n_r * \frac{a - \min(A,r)}{\max(A,r) - \min(A,r)}$$

  - If expression is part of query, value of v may not be available.
    - Then rough estimate is $n_r / 2$

Selection size estimation

- $\sigma_{\theta 1 \wedge \theta 2 \ldots \wedge \theta n}(r)$
  - For each i if the estimate of $\sigma_{\theta i}(r)$ is s_i then probability of a tuple satisfying θ_i is s_i / n_r
  - Estimate $$n_r * \frac{s_1 * s_2 * \ldots s_n}{n_r^{nr}}$$
- $\sigma_{\theta 1 \vee \theta 2 \ldots \vee \theta n}(r)$
  - Estimate $$n_r * (1 - (1 - s_1/n_r)(1 - s_2 / n_r) \ldots (1 - s_n / n_r))$$
- $\sigma_{\neg \theta}(r)$ Exercise

Join size estimation

- Cartesian product $r \times s$ will have $n_r * n_s$ tuples
- Natural Join : Let R and S be the set of attributes of r and s respectively
  - If $R \cap S = \varnothing$ then Natural Join is same as cartesian product
  - If $R \cap S$ is key for R then we can have at most n_s tuples in the natural join
  - If $R \cap S$ is a key neither for R nor for S if $R \cap S = \{ A \}$ then
    - Every tuple t in R produces $n_s / V(A,s)$ many tuples
    - So total estimate is $n_r * n_s / V(A,s)$
    - Reversing the roles of r and s, we get the estimate $n_r * n_s / V(A,r)$
      - Estimates differ if $V(A,r) \neq V(A,s)$ then pick the minimum
- We can also estimate $r \bowtie_\theta s$ as $\sigma_\theta (r \times s)$