Fundamentals of Database Systems

[Query Optimization – I]

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
Indian Statistical Institute, Kolkata

September, 2019

Basics of query optimization

Why optimizing a query?

So that it is processed efficiently.

Basics of query optimization

Why optimizing a query?
So that it is processed efficiently.

What is meant by efficiently?
Minimizing the cost of query evaluation.

Basics of query optimization

Why optimizing a query?
So that it is processed efficiently.

What is meant by efficiently?
Minimizing the cost of query evaluation.

Who will minimize?
The system, not the user.

Basics of query optimization

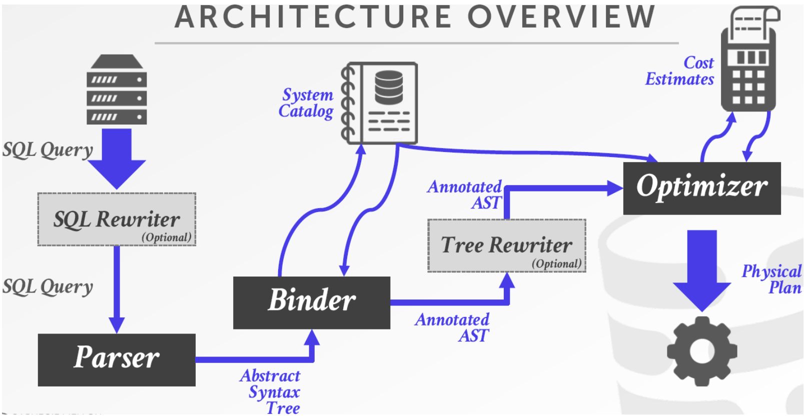
Why optimizing a query?
So that it is processed efficiently.

What is meant by efficiently?
Minimizing the cost of query evaluation.

Who will minimize?
The system, not the user.

Query optimization is facilitating a system to construct a query-evaluation plan for processing a query efficiently, without expecting users to write efficient queries

How does a query optimizer work?

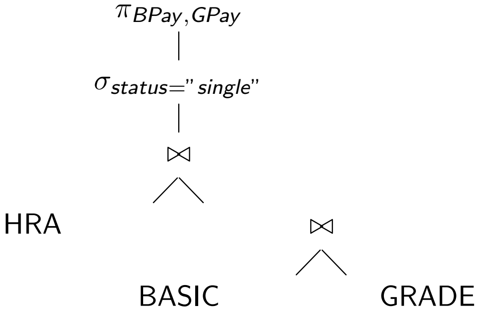


An example query

Find the basic pay and grade pay of ISI employees who are single.

An example query

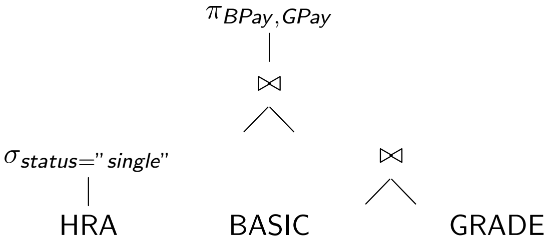
Find the basic pay and grade pay of ISI employees who are single.



Note: Marital status is a part of HRA table.

An example query – revised

Find the basic pay and grade pay of ISI employees who are single.



Note: Marital status is a part of HRA table.

Evaluating query cost

The basic parameters for estimating the query cost are

- The number of seek operations performed
- The number of blocks read
- The number of blocks written

Evaluating query cost

The basic parameters for estimating the query cost are

- The number of seek operations performed
- The number of blocks read
- The number of blocks written

Intuitive (Naive) approach for least-cost query finding:

1. Generate expressions that are logically equivalent to the original expression
2. Annotate the resultant expressions in alternative ways to generate alternative query evaluation plans.
3. Go to 1 until you get some new expression.

Catalog information

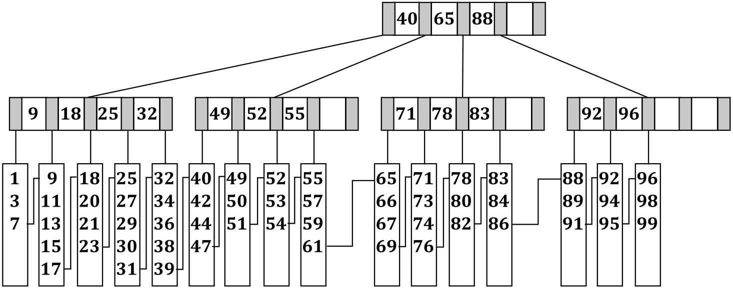
For a given relation R, we can store the following relevant information in the catalog:

- N_R – the number of tuples
- B_R – the number of blocks containing tuples of relation R
- L_R – the size of a tuple in bytes
- F_R – the number of tuples that fit into one block (blocking factor)
- $V(X, R)$ – the number of distinct values for attribute X
- H_R – the height of B$^+$-tree indices for R
- L_R – the number of leaf pages in the B$^+$-tree indices for R

Note: $V(X, R)$ equals to the size of $\pi_X(R)$, in general, and if X is a key then it is N_R.

The storage of data in B^+-Trees

The B^+ tree is a balanced binary search tree that follows a multi-level index format. It can support both random access and sequential access of the data items.

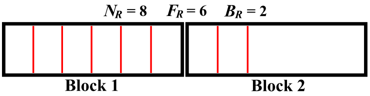


A B^+-Tree of order 5 and depth 3 consisting of 59 data items

Other statistical information

Suppose the tuples of a relation R are physically stored in a file then we have the following relation

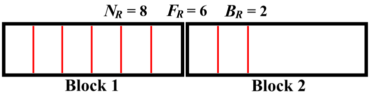
$$B_R = \left\lceil \frac{N_R}{F_R} \right\rceil$$



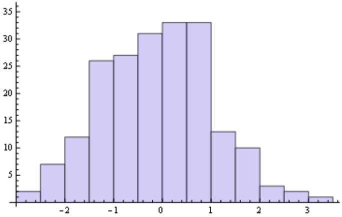
Other statistical information

Suppose the tuples of a relation R are physically stored in a file then we have the following relation

$$B_R = \left\lceil \frac{N_R}{F_R} \right\rceil$$



Special statistical information:

Histogram – 

Final comments

Some facts about the relation statistics:

– Recompute relation statistics on every update (but this might be a huge overhead), at least during the periods of light system load.

– In real-world cases, optimizers often maintain further statistical information to improve the accuracy of their cost estimates of evaluation plans.

Size estimation for selection operation

Assumption: Attribute values are uniformly distributed

- $\mathcal{S}(\sigma_{X=x}(R))$: $\frac{N_R}{V(X,R)}$
- $\mathcal{S}(\sigma_{X\leq x}(R))$: $\frac{N_R*(x-\min(X,R))}{\max(X,R)-\min(X,R)}$, where $\min(X,R)$ and $\max(X,R)$ denote the minimum and maximum values of the attribute X in R, respectively
- $\mathcal{S}(\sigma_{\theta_1\wedge\theta_2\wedge\ldots\wedge\theta_n}(R))$: $\frac{\mathcal{S}_1*\mathcal{S}_2*\ldots*\mathcal{S}_n}{N_R^{n-1}}$, where $\mathcal{S}_i$ denotes the estimated size of the selection operation $\sigma_{\theta_i}(R)$
- $\mathcal{S}(\sigma_{\theta_1\vee\theta_2\vee\ldots\vee\theta_n}(R))$: $\frac{N_R^n-(N_R-\mathcal{S}_1)*(N_R-\mathcal{S}_2)*\ldots*(N_R-\mathcal{S}_n)}{N_R^{n-1}}$, where $\mathcal{S}_i$ denotes the estimated size of the selection operation $\sigma_{\theta_i}(R)$
- $\mathcal{S}(\sigma_{\neg\theta}(R))$: $N_R - \mathcal{S}(\sigma_\theta(R))$

Note: A predicate is expressed as θ.

Size estimation for selection – A conceptual example

Consider the following table and its system catalog information:

Table: TOY

ID	COLOR	COST
T1	Blue	10
T2	Blue	10
T3	Red	20
T4	Blue	20
T5	Red	30
T6	Red	30

$N_{TOY} = 6$
$V(COLOR, TOY) = 2$
$V(COST, TOY) = 3$
$\min(COST, TOY) = 10$
$\max(COST, TOY) = 30$
$H_{TOY} = 3$

- $\mathcal{S}(\sigma_{COLOR=Red}(TOY))$: $\frac{N_{TOY}}{V(COLOR,TOY)} = 3.$
- $\mathcal{S}(\sigma_{COST \leq 20}(TOY))$: $\frac{N_{TOY} * (20 - \min(COST,TOY))}{\max(COST,TOY) - \min(COST,TOY)} = 3.$
- $\mathcal{S}(\sigma_{(COLOR=Red) \wedge (COST \leq 20)}(TOY))$: $\frac{\mathcal{S}(\sigma_{COLOR=Red}(TOY)) * \mathcal{S}(\sigma_{COST \leq 20}(TOY))}{N_{TOY}} = 1.5$

Size estimation for Cartesian product and natural join

Cartesian product:

$\mathcal{S}(R_1 \times R_2)$ is equal to $N_{R_1} * N_{R_2}$ (each tuple occupies $L_{R_1} + L_{R_2}$ bytes).

Natural join:

- If $A(R_1) \cap A(R_2) = \phi$: $\mathcal{S}(R_1 \bowtie R_2)$ equals to $N_{R_1} * N_{R_2}$
- If $A(R_1) \cap A(R_2)$ is a key for R_1: $\mathcal{S}(R_1 \bowtie R_2)$ is no greater than N_{R_2}
- If $A(R_1) \cap A(R_2)$ is a key for R_2: $\mathcal{S}(R_1 \bowtie R_2)$ is no greater than N_{R_1}
- If $A(R_1) \cap A(R_2)$ is a key for neither R_1 nor R_2: $\mathcal{S}(R_1 \bowtie R_2)$ is the maximum of $\frac{N_{R_1} * N_{R_2}}{V(X,R_1)}$ and $\frac{N_{R_1} * N_{R_2}}{V(X,R_2)}$

Size estimation for other operations

- Projection: $\mathcal{S}(\pi_X(R))$ equals to $V(X,R)$
- Aggregation: Involves a size of $V(X,R)$
- Set union operation: $\mathcal{S}(R_1 \cup R_2)$ is no greater than $N_{R_1} + N_{R_2}$
- Set intersection operation: $\mathcal{S}(R_1 \cap R_2)$ is no greater than $\min(N_{R_1}, N_{R_2})$
- Set difference operation: $\mathcal{S}(R_1 - R_2)$ is no greater than N_{R_1}

Query size estimation – Example I

Given a relation R with 60 tuples. If R has an attribute Age within the range [20, 30] and there are 15 distinct values for the attribute $Height$ minimum of which is 170, estimate the size of the query $\sigma_{(Age \leq 23) \vee Height = 170}(R)$.

Query size estimation – Example I

Given a relation R with 60 tuples. If R has an attribute Age within the range [20, 30] and there are 15 distinct values for the attribute $Height$ minimum of which is 170, estimate the size of the query $\sigma_{(Age \leq 23) \vee Height = 170}(R)$.

Solution: It is given that $N_R = 60$, $\min(Age, R) = 20$, $\max(Age, R) = 30$, $\min(Height, R) = 170$ and $V(Height, R) = 15$. Therefore, with uniform distribution assumption, the size of the query can be estimated as $\mathcal{S}(\sigma_{(Age \leq 23) \vee Height = 170}(R))$

$$= \frac{N_R^2 - (N_R - \mathcal{S}(\sigma_{Age \leq 23}(R))) * (N_R - \mathcal{S}(\sigma_{Height=170}(R)))}{N_R}$$

$$= \frac{N_R^2 - (N_R - \frac{N_R * (23 - \min(Age,R))}{\max(Age,R) - \min(Age,R)}) * (N_R - \frac{N_R}{V(Height,R)})}{N_R}$$

$$= 20.8.$$

Query size estimation – Example II

Let $R1(ID, Name)$ **and** $R2(Roll, CGPA)$ **be a pair of relations. Now if** ID **be the primary key for** $R1$ **and the attribute** $Roll$ **has a minimum value of 118002001, then estimate the size of the query** $\sigma_{ID=11}(R1) \bowtie \sigma_{Roll \leq 118002001}(R2)$**.**

Query size estimation – Example II

Let $R1(ID, Name)$ and $R2(Roll, CGPA)$ be a pair of relations. Now if ID be the primary key for $R1$ and the attribute $Roll$ has a minimum value of 118002001, then estimate the size of the query $\sigma_{ID=11}(R1) \bowtie \sigma_{Roll \leq 118002001}(R2)$.

<u>Solution</u>: If ID be the primary key of $R1$, then it should have distinct values satisfying $V(ID, R1) = N_{R1}$. Again it is given that $\min(Roll, R2) = 118002001$. Therefore, with the assumption of uniform distribution over the attribute domains, the given query size can be estimated as $\mathcal{S}(\sigma_{ID=11}(R1) \bowtie \sigma_{Roll \leq 118002001}(R2))$

$$
\begin{aligned}
&= \mathcal{S}(\sigma_{ID=11}(R1)) \times \mathcal{S}(\sigma_{Roll \leq 118002001}(R2)) \\
&= \frac{N_{R1}}{V(ID, R1)} \times \frac{N_{R2} * (118002001 - \min(Roll, R2))}{\max(Roll, R2) - \min(Roll, R2)} \\
&= \frac{N_{R1}}{N_{R1}} \times \frac{N_{R1} * (118002001 - 118002001)}{\max(Roll, R2) - 118002001} = 0.
\end{aligned}
$$

Problems

1. Given a pair of relations $R1$ and $R2$, wherein the primary key of $R1$ (say K) is the only foreign key of $R2$, estimate the size of the following queries. Assume that the number of tuples in $R1$ and $R2$ are $t1$ and $t2$, respectively.
   i. $R1 \bowtie R2$.
   ii. $R1 \times R2$.
   iii. $\sigma_{K='19BM6JP01'}(R1) \bowtie R2$.