WEEK 9

QUERY OPTIMIZATIONS – INTRODUCTION

STUDENT OBJECTIVES

- Upon completion of this video, you should be able to:
  - Identify situations where performing your query in a different order will change the speed with which it is executed.
  - List 9 strategies you can use to optimize a query
  - Given a query, find the estimate of selectivity for it.

INTRODUCTION AND MOTIVATION

Suppose we need to answer the following query:

Give me all the last names of employee's who work in the Research Department.

In SQL we would write:

SELECT lastname FROM employee e, department d WHERE e.deptnum = d.deptnum AND deptname = "Research Department"

Which would get translated to the following relational algebra:

$\pi_{\text{LastName}}(\sigma_{\text{DeptName = "Research"}}(\text{Employee} \bowtie \text{Department}))$

Or

projection

$\pi_{\text{LastName}}(\sigma_{\text{e.DeptNum = d.DeptNum and DeptName = "Research"}}(\text{Employee X Department}))$

cartesian product.

EMPLOYEE

SSN	FirstName	LastName	DeptNum
1	Laura	Reid	PR
2	Sue	Jones	PR
3	Colleen	Smith	IS
5	Brenda	Jones	RS
7	Doug	Vancise	IS
8	Sandra	Dillon	IS
9	Stephen	Watt	RS
10	Sue	Smith	CS
12	Janice	Jones	RS
15	Sandra	Dillon	CS
17	Jamie	Andrews	CS
22	Janice	Dillon	IS
23	Peter	Aziz	IS
29	Scott	Aziz	CS

DEPARTMENT

DeptId	DeptName
RS	Research
PR	Payroll
IS	Information Services
CS	Customer Service

QUESTION 1: What names result from our query?

LastName
Jones
Watt
Jones

← this line may not show

QUESTION 2: How did you do it so quickly?

Examine the following 2 ways to execute this query:

One:

-Join tables Employee and Department with a Cartesian Product and then done a join to eliminate some rows.

-From the temporary table produced, select out only the “Research Department” ones

Two:

-Find the “Research Department” Department Number

-Go through the Employee table looking for that Dept Number

QUESTION: Which method is faster? One or Two

QUERY OPTIMIZATION

- Consider the following select statement

SELECT student.sid FROM student, enroll WHERE student.sid = enroll.sid AND enroll.mark > 95

- Assume:
  - 10,000 tuples in the enroll table, each tuple is 25 char (each char is 1 byte in ASCII)
  - 2% of enroll table students have a mark over 95 %,
  - sid (joining attribute) is 10 bytes, mark is 4 bytes
  - 1000 tuples in the student table, each tuple is 100 char

Suppose we try to answer the above query using 2 different methods:

Option 1: bring everything from disk and then do join:

$$\pi\ sid(\sigma\ mark>95(STUDENT \bowtie ENROLL))$$

row bytes.

This results in 250,000 bytes (10000 * 25) + 100000 bytes (1000*100) = **350,000 bytes read**

Option 2: bring just what you need from disk and then do join:

$$\pi\ sid((\pi\ sid(STUDENT) \bowtie \pi\ sid(\sigma\ mark>95(ENROLL))$$

This results in 2800bytes (0.02*10000*14) + 10000bytes (10*1000) = 2800 +10000 = **12,800 bytes read**

MORAL: The order in which we do our SQL operations can GREATLY effect the speed with which the query is generated!

THE ORDER WE PERFORM OPERATIONS IN A QUERY CAN DRASTICALLY AFFECT THE TIME TO RETURN THE RESULTS TO THE QUERY, THUS RELATIONAL DATABASE MANAGEMENT SYSTEMS PROVIDE A: *QUERY OPTIMIZER.*

In the network and hierarchical model the programmer had to manually optimize the queries.

NOTE: The DBMS is not really **optimizing (best solution)** more like finding a reasonably efficient strategy. It is called finding a query execution plan.

but not the best one.

Question: Why not always find the optimal solution?

Find a best solution also takes a long time.

- The query optimizer breaks down a query into query blocks (usually it translates the blocks into relational algebra) and then chooses an execution plan for each block.

- What methods does the DBMS use to speed up the processing and optimize and execute queries?
  - The query is **scanned** (finds the tokens in the queries), **parsed**(checks grammar of query language),
  - **validated** (checks that all attributes and relation names are valid),
  - then a **query tree** is created to determine an execute strategy for retrieving the data
  - the query tree is **optimized**
  - the best algorithm is chosen for each operation

SELECT OPTIMIZING STRATEGIES

- Assume we have our 5 standard table from our big example:
  - **Employee:** This table is sorted by the department number and has a clustering index on the department number. It also has a secondary index on the last name of the employees.
  - **Department:** This table has is sorted on the department number and has a primary index on the department number.
  - **Project:** This table has used the project number as a hash key.
  - **WorksOn:** Composite index on SSN and ProjNumber
  - **Dependent:** This table is not ordered and has no indices

SELECT OPTIMIZING STRATEGIES

- Consider the following implementations for a SELECT operation:
  - →1. Linear Search (worst)
  - →2. Binary Search
  - →3. Primary Key Index (B+ tree or hash directly)
  - →4. Range Search (B+ tree)
  - →5. Clustering Index (searching on a non-key attribute, equality search with a clustered index)
  - →6. Secondary Index on Field

- Selects with multiple conditions, for example:
  (Age > 25 and Sex = 'M'):
  - →7. **Conjunctive Select:** if one of the attributes is a key use options 2 to 6 from above on it first, then test the remaining conditions on the resulting records
  - →8. **Conjunctive Select** using a composite index, if 2 or more attributes are part of a composite key use the index directly (**WorksOn** might have a composite key on SSN and ProjNumber)
  - →9. **Conjunctive Select** by intersection of record pointers: secondary indexes on individual records, not blocks on more than one attribute, look for the intersection of the record pointers.
  - →**Disjunctive Select:** (Age > 25 **OR** DNO = 5 **OR** Sex='F'), not much can be done here to optimize but try to use access paths (indexes)

OR is costly.

QUESTION: Consider the following queries and decide which one from above is appropriate:

SELECT * FROM employee WHERE ssn = 123 1

SELECT * FROM department WHERE deptno > 5 3

SELECT * FROM employee WHERE lastname = ‘Simpson’ 6

SELECT * FROM project WHERE pnumber = 34 3

SELECT * FROM employee WHERE deptno=5 AND salary > 3000 AND sex = ‘F’ AND lastname > ‘M’ 5, 6

SELECT * FROM department WHERE deptno > 5 AND deptname > ‘M’ 3,4,7

SELECT * FROM works_on WHERE essn=123 AND pno=10 8

SELECT * FROM dependent WHERE essn=123 1

ESTIMATES OF SELECTIVITY

- Estimates of Selectivity: the smaller the estimate is, the more **desirable** to use that select first (we will use this in the query optimize tree later on!)
- S=ratio of tuples to satisfy a condition to the total number of record in a relation (always gives a number between 0 and 1)
- Examples:

at worst

query made.

  - *Select … WHERE key = 34*, assume there are 100 tuples, then you would get S=1/100 = 0.01 great to perform first.
  - *Select … WHERE sex = 'M'*, assume there are 100 tuples and 50 males, you would get S= 50/100 = 0.5

closer to 1, worse.