

# CS3743 Introduction to Database Systems Query Evaluation and Optimization

Dr. Weining Zhang

Department of Computer Science  
University of Texas at San Antonio

October 22, 2009

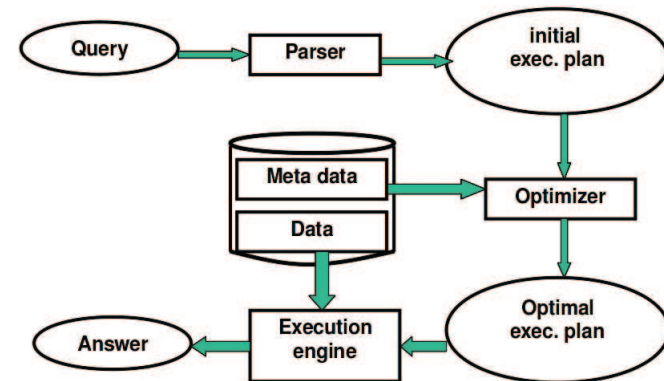
## Outline

- 1 Query Evaluation and Optimization
  - Overview of the Process
  - Query Optimization
  - Cost Estimation

## Introduction

- DBMS let users to specify queries in declarative languages, such as SQL. It must then determine the best evaluation plan for each query.
- Query processor needs to
  - represent evaluation plans in some form (typically a tree or a graph)
  - estimate cost (in terms of page I/Os) of each plan
  - explore the space of different evaluation plans to find the most efficient plan (in theory) and to avoid the worst plan (in practice)

## Query Evaluation Process



# Types of Query Evaluation Plans

- **Logic Plan.** A query tree representing a relational algebra expression
  - Non-leaf nodes are relational operators
  - Leaf nodes are relations
- **Physical Plan.** An annotated query tree
  - including additional operations not in the algebra
  - operations are annotated by specific algorithms using fast access paths

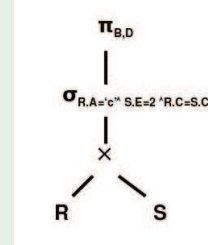
# Example Logic Evaluation Plans

## Example

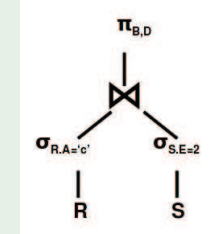
Select B,D From R,S

Where R.A = 'c' and S.E = 2 and R.C=S.C

$$\Pi_{B,D}(\sigma_{R.A='c' \wedge S.E=2 \wedge R.C=S.C}(R \times S)) \quad \Pi_{B,D}(\sigma_{R.A='c'}(R) \bowtie \sigma_{S.E=2}(S))$$



Plan 1

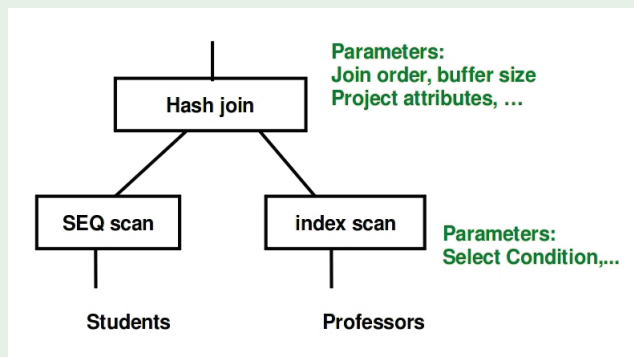


Plan 2

# Example Physical Plan

## Example

select Name from Students, Professors  
where Advisor=PID and Dept='Computer Science'



# Query Optimization

- Logic query optimization
  - Use algebra laws to transform initial logic plan into a more efficient logic plan
  - Since the optimization problem is NP-hard, heuristics are used for logic query optimization
- Cost-based Optimization
  - Estimate cost of a query plan (or a part of a query plan) based on data statistics, algorithms, and access paths
  - Use estimated costs to limit the search in the space of query plans, and to avoid the most expensive query plans
- The selected query plan can be
  - executed directly (by interpreter)
  - compiled and executed late (using a compiler)

## Heuristic Query Optimization

- Query trees can be transformed according to rules for relational algebra operations
- There are many transformation rules. For example
  - Communicating  $\sigma$  with  $\bowtie$ :  $\sigma_C(R \bowtie S) \equiv (\sigma_C(R)) \bowtie S$  (push selection over join)
  - Converting a  $(\sigma, \times)$  sequence into  $\bowtie$ :  $\sigma_C(R \times S) \equiv R \bowtie_C S$
  - Associativity of  $\bowtie$ :  $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$  (change the join order)
- Rules of heuristic query optimization
  - Push selections towards leaves as much as possible
  - Replace cross products by joins
  - Reorder joins to perform selective joins earlier
  - Push projection towards leaves

## Cost-based Query Optimization

- The goal is to find a physical plan from a good logical plan, so that it has the minimal estimated I/O cost
  - Generate physical evaluation plans from a logic plan by assigning algorithms to nodes in plan tree
  - Estimate I/O costs of algebra operations while generation of physical plans
  - Estimate I/O costs of subtree based on estimated costs of plan nodes
  - Abandon plans or partial plans whose costs are not minimum
    - There are many heuristic algorithms for physical plan enumeration

## Cost of Query Evaluation

- Cost = I/O cost + CPU cost
  - I/O cost: number of pages read and written
  - CPU cost: number of comparisons or tuples processed
- I/O cost dominates the query cost and is the focus of query optimization
- I/O cost is estimated according to
  - statistics (the meta data) maintained by the DBMS
  - query conditions of specific relational algebra operations
  - specific algorithms of algebra operations

## Data Statistics

- Statistics of tables and attributes are stored in database, and used by query optimizer
- Let  $R$  and  $S$  be relations. Following statistics are used in cost models.
  - $n_R$ : the number of tuples in  $R$
  - $b_R$ : number of pages in  $R$
  - $bf_R$ : blocking factor of  $R$
  - $dist(R.A)$ : number of distinct values in column  $R.A$
  - $min(R.A)$  and  $max(R.A)$ : the smallest and the largest values in  $R.A$
  - $HI$ : height of a B+ tree index or the number of index pages to access

## Cost of Selection

- Consider a simple selection  $\sigma_{A\theta a}(R)$ .
  - Number of selected tuples:  $NS = n_R \times sf_{A\theta a}(R)$ , where  $0 \leq sf_{A\theta a}(R) \leq 1$  is the selectivity factor
- Linear Search. Cost depends on if  $A$  is a key and if  $\theta$  is equality
  - If the condition is equality on a key, average number of I/O pages is  $b_R/2$
  - the cost is  $b_R$  for general condition
- Binary Search. Records must be sorted
  - Equality on key:  $cost = \lceil \log_2(b_R) \rceil$
  - Equality on non-key:  $cost = \lceil \log_2(b_R) \rceil + \lceil NS/bf_R \rceil + 1$

## Cost of Join

- Consider a simple join  $R \bowtie_{A\theta B} S$ .
  - Number of tuples in join result:  $NJ = \frac{n_R \cdot n_S}{\max(dist(R.A), dist(S.B))}$
- There are several join algorithms
  - Blocked Nested Loop Join (BNLJ):
    - $R$  as outer relation:  $cost = b_R + b_R \cdot b_S + \lceil NJ/bf_{result} \rceil$
    - $S$  as outer relation:  $cost = b_S + b_R \cdot b_S + \lceil NJ/bf_{result} \rceil$
  - Merge Join
  - Hash Join

## Cost of Selection (cont.)

- Index Search. If index is available on the search attribute, search the index for matching rids; then access data file to retrieve records of the rids
  - Cost = cost of searching index + cost of retrieving data
    - Equality on primary index:  $Cost = HI + 1$
    - Equality on clustering index:  $Cost = HI + \lceil NS/bf_R \rceil$
    - Equality on secondary index:  $Cost = HI + NS$

## Example Selection Costs

Consider relation  $R(A, B, C)$ . Assume  $n_R = 10,000$  tuples,  $bf_R = 20$  tuples/page,  $dist(A) = 50$ , and  $dist(B) = 500$ . There is an order 25 B+ tree clustering index on  $R.A$ , and an order 25 B+ tree secondary index on  $R.B$ . Estimate the I/O cost of  $\sigma_{A=a \wedge B=b}(R)$ .

- $S_1$ : Sequential Scan
  - $Cost = b_R = \lceil 10,000/20 \rceil = 500$
- $S_2$ : Binary Search using  $A = a$  (We know that  $R$  is sorted on  $R.A$ . Why?)
  - $NS = 10000/50 = 200$
  - $Cost = \lceil \log_2(b_R) \rceil + \lceil NS/bf_R \rceil - 1 = \lceil \log_2(500) \rceil + \lceil 200/20 \rceil - 1 = 18$

## Example Selection Costs (cont.)

- $S_3$ : Use index on  $R.A$ 
  - $HI = 2$  because average order of B+ tree is  $(P + 0.5P)/2 = 19$  (i.e., leaf nodes have 18 entries, internal nodes have 19 pointers). Thus, there are  $\lceil 50/18 \rceil = 3$  leaves and 1 node in the next level
  - $Cost = HI + \lceil NS/bf_R \rceil = 2 + \lceil 200/20 \rceil = 12$
- $S_4$ : Use index on  $R.B$ 
  - $HI = 4$  because average order is 19;  $\lceil 10000/18 \rceil = 556$  nodes in leaf level;  $\lceil 556/19 \rceil = 29$  nodes in level 2; 2 nodes at level 3 and 1 node at level 4.
  - $Cost = HI + NS = 4 + 20 = 24$  where  $NS = 10000/500 = 20$  assuming Option 1 of secondary index (allowing duplicate keys)