Introduction to Database Systems

CSE 444

Lecture #17
March 7, 2001

Announcements

- Today
  - HW#4 (optional) due
  - Project Interviews
    - 5-7pm today, 3-5pm tomorrow
- HW Solutions
  - HW#3 available later today
  - HW#4 available by Friday
- Graded Homework
  - HW#3 – being handed back
  - HW#4 – available next Tuesday
- Resolving Grading Issues
  - HW#3 grading issue – must resolve by this Friday
  - HW#4 grading issue – must resolve before exam

2

Announcements

- Teaching Evaluation
  - Need a volunteer!
- Different office hours for next week
  - Yana: M, W 4-5pm
  - Yana: appointment via email for Thu 8-10am
  - Surajit: Tu 1.00-2.00pm
  - Surajit: Thu 11.30am-12.30pm **Office: 226D**
- Continue to check mail archive for all communications/issues related to the course
- Final Exam Thu March 15
  - In Class (Loew 102) 2.30-4.20pm
  - 100 points, 100 mins
  - Questions – 5 to 10 points each

3

Selective Exclusion from Finals

- All concepts included.
- Exclusions only apply to "direct questions"
  - No direct questions on E-R Diagrams (Chapter 2, 3.1-3.4)
    - But, concepts of relationships, key, constraints included
  - No direct questions on deadlocks (Chapter 10.3 – Vol 2)
  - No direct questions on Media Failure (Chapter 8.5- Vol 2)
  - No direct questions from "Undo-only" and "Redo-only" logging (Chapter 8.3-8.4 Vol 2) but there will be questions from "Undo/Redo Logging" (Chapter 8.5 Vol 2)
    - Be familiar with concepts in Chapters 8.3-8.4 (Vol 2)

4

Query Optimization

Required Reading: 7.2, 7.4, 7.5, 7.6, 7.7.0 – 7.7.2, 7.7.6

Query Optimization: Phases

- Parsing phase
  - Produces a parse tree
- Query-Rewrite phase
  - Produces a logical tree
- Physical Query plan generation
  - Produces executable (physical) plan

6

Query Optimization

- Algebraic laws provide alternative execution plans
- Estimate costs of alternative modes of execution
- Efficiently search the space of alternatives
  - Simplify search by applying heuristics (without costing)
    - apply laws that *seem* to result in cheaper plans

7

Converting from SQL to Logical Plans

Select a1, …, an
From R1, …, Rk
Where C

$\Pi_{a1,\ldots,an}(\sigma_C(R1 \bowtie R2 \bowtie \ldots \bowtie Rk))$

8

Enumerating Physical Plans

- Exhaustive – Consider all possible:
  - Join Orders
  - Algorithms for each operator
- Heuristic Search
  - E.g. Greedy approach
  - Pick next relation such that join size is smallest

9

Enumerating Physical Plans

- Branch-and-Bound Enumeration
  - Find a good starting plan (having cost C)
  - In subsequent search, eliminate any subquery with cost > C
- Hill Climbing
  - Start with heuristically selected plan
  - Explore plans in the "neighborhood"
    - E.g. replace Nested-Loops join with Hash-Join

10

Enumerating Physical Plans

- Dynamic Programming
  - Bottom-up strategy
  - For each subexpression, only keep plan with the least cost
    - Consider possible implementations of each node assuming
  - Must consider *interesting orders*
    - E.g., when subexpression is sorted on a sort attribute at the node
  - More later

11

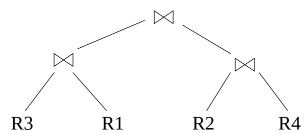
Determining Join Order

- Select-project-join
- Push selections down, pull projections up
- We need to choose the join order
- This is the main focus of our study today

12

Determining Join Order: Join Trees

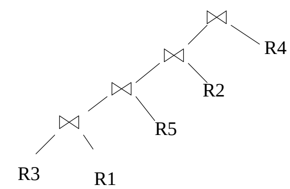
- R1 ⋈ R2 ⋈ .... ⋈ Rn
- Join tree:



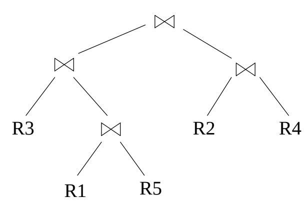
- A join tree represents a plan. An optimizer needs to inspect many (all ?) join trees

Linear Join Trees

- Left deep:



Bushy Join Trees



Join Ordering Problem

- Given: a query R1 ⋈ R2 ⋈ … ⋈ Rn
- Assume we have a function cost() that gives us the cost of every join tree
- Find the best *linear join* tree for the query

Intro to Enumeration using Dynamic Programming

- For each subquery Q ⊆ {R1, …, Rn} compute the following:
  - Size(Q)
  - A best plan for Q: Plan(Q)
  - The cost of that plan: Cost(Q)

Dynamic Programming (1)

- Step 1: For each {Ri} do:
  - Size({Ri}) = B(Ri)
  - Plan({Ri}) = Ri
  - Cost({Ri}) = (cost of scanning Ri)

Dynamic Programming (2)

- Step i: For each Q ⊆ {R1, …, Rn} of cardinality i do:
  - Compute Size(Q)
  - For every subquery Q' and a relation Rj s.t. Q = Q' U {Rj} compute cost(Plan(Q') ⋈ Rj)
  - Cost(Q) = the smallest such cost
  - Plan(Q) = the corresponding plan
- Final Step: Return Plan({R1, …, Rn})

19

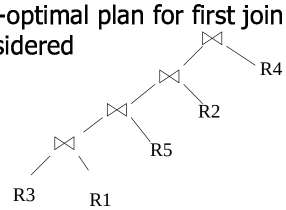
Comments on Enumeration

- Recall: computes optimal plans for subqueries:
  - Step 1: {R1}, {R2}, …, {Rn}
  - Step 2: {R1, R2}, {R1, R3}, …, {Rn-1, Rn}
  - …
  - Step n: {R1, …, Rn}
- Read Example 7.3.5 (important)
- Practical Issues
  - Heuristics for Reducing the Search Space
    - Restrict to trees "without cartesian product"
  - Need more than just one plan for each subquery:
    - "interesting orders"

20

Role of Interesting Order

- Join conditions: R1.a = R3.a = R5.a
- Sub-optimal plan for first join need to be considered



21

Completing the Physical Query Plan

- Choose algorithm to implement each operator
  - Need to account for more than cost:
    - How much memory do we have ?
    - Are the input operand(s) sorted ?
- Decide for each intermediate result (not included in exam):
  - To materialize or to pipeline

22

Choice of Algorithms: Selection

- Sequential scan
- Index-based
  - Identify selection condition for which an index exists
  - Retrieve all tuples using the index
    - How many are there?
    - How much does it cost?
      - Equality condition – easy
      - Inequality condition:
        - Clustered Index: B (R) /3, Non-clustered Index: T(R)/ 3
  - Post-Filter other predicates
- Read Example 7.37

23

Choice of Algorithms: Join

- One pass algorithms works for "small" sizes)
- Sort-Join
  - When an order exists at least partially
  - Multiple joins on same attribute (interesting order)
- Index Nested Loop Join
  - Index on inner
- Hash Join

24

Physical Query Plan

⌘Leaves
- ⍓Tablescan®, Indexscan(R,C), Indexscan (R,A)

⌘Selection
- ⍓Fiter©, Indexscan(R,C)

⌘Join
- ⍓Hash-Join(), ..

⌘Sort()….

25

What did you learn?

⌘Database schema, its design principles

⌘SQL – Writing Database Programs

⌘Connectivity/Client Server Issues

⌘Inside DBMS
- ⍓Transactions, Storage, Query Engine

⌘We did not cover..
- ⍓Data Analysis, Warehouse, Mining
- ⍓TP Monitor (Application Server)
- ⍓XML
- ⍓Multimedia

26