


Query Processing


1/14/2005 Yan Huang - CSCIS330 Database
Implementation - Query Processing



Hybrid Merge-join

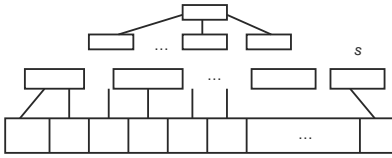
- If one relation is sorted, and the other has a secondary B⁺-tree index on the join attribute
 - Merge the sorted relation with the leaf entries of the B⁺-tree .
 - Sort the result on the addresses of the unsorted relation's tuples
 - Scan the unsorted relation in physical address order and merge with previous result, to replace addresses by the actual tuples
 - Sequential scan more efficient than random lookup

1/14/2005 Yan Huang - CSCIS330 Database
Implementation - Query Processing



Hybrid Merge-join


r



1/14/2005 Yan Huang - CSCIS330 Database
Implementation - Query Processing

Exercise

- Compute $depositor \bowtie customer$, with $depositor$ as the outer relation.
- $customer$ has a secondary B⁺-tree index on $customer-name$
 - Blocking factor 20 keys
- $\#customer = 400b/10,000t$ $\#depositor = 100b/5,000t$
- Merge join
 - $\log(400) + \log(100) + 400 + 100 = 516$



1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Hash join

- Hash function h , range $0 \rightarrow k$
- Buckets for R1: G_0, G_1, \dots, G_k
- Buckets for R2: H_0, H_1, \dots, H_k

Algorithm

- Hash R1 tuples into G buckets
- Hash R2 tuples into H buckets
- For $i = 0$ to k do
 - match tuples in G_i, H_i buckets

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Simple example hash: even/odd

R1	R2	Buckets	
2	5	Even: 2 4 8	4 12 8 14
4	4		
3	12	Odd: 3 5 9	5 3 13 11
5	3		
8	13		
9	8		
	11		
	14		

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Minimum memory requirements:

Size of R1 bucket = (x/k)

k = number of memory buffers
 x = number of R1 blocks

So... $(x/k) < k$

$k > \sqrt{x}$

Which relation should be the build relation?

1/14/2005 Yan Huang - CSC5330 Database Implementation - Query Processing

Example of Cost of Hash-Join

customer \bowtie *depositor*

- Assume that memory size is 20 blocks
- $b_{depositor} = 100$ and $b_{customer} = 400$.
- depositor* is to be used as build input. Partition it into five partitions, each of size 20 blocks. This partitioning can be done in one pass.
- Similarly, partition *customer* into five partitions, each of size 80. This is also done in one pass.
- Therefore total cost: $3(100 + 400) = 1500$ block transfers
 - ignores cost of writing partially filled blocks

1/14/2005 Yan Huang - CSC5330 Database Implementation - Query Processing

Complex Joins

- Join with a conjunctive condition:

$$r \bowtie_{\theta_1 \wedge \theta_2 \wedge \dots \wedge \theta_n} s$$
 - Either use nested loops/block nested loops, or
 - Compute the result of one of the simpler joins $r \bowtie_{\theta_i} s$
 - final result comprises those tuples in the intermediate result that satisfy the remaining conditions
$$\theta_1 \wedge \dots \wedge \theta_{i-1} \wedge \theta_{i+1} \wedge \dots \wedge \theta_n$$
- Join with a disjunctive condition

$$r \bowtie_{\theta_1 \vee \theta_2 \vee \dots \vee \theta_n} s$$
 - Either use nested loops/block nested loops, or
 - Compute as the union of the records in individual joins $r \bowtie_{\theta_j} s$:

$$(r \bowtie_{\theta_1} s) \cup (r \bowtie_{\theta_2} s) \cup \dots \cup (r \bowtie_{\theta_n} s)$$

1/14/2005 Yan Huang - CSC5330 Database Implementation - Query Processing

Other Operations

- **Duplicate elimination** can be implemented via hashing or sorting.
- **Projection** is implemented by performing projection on each tuple followed by duplicate elimination.

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Other Operations : Aggregation

- **Aggregation**
 - Sorting
 - Hashing

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Other Operations : Set Operations

- **Set operations** (\cup , \cap and \rightarrow)
 - can either use variant of merge-join after sorting
 - or variant of hash-join.

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Evaluation of Expressions



- So far: we have seen algorithms for individual operations
- Alternatives for evaluating an entire expression tree
 - **Materialization**: generate results of an expression whose inputs are relations or are already computed, **materialize** (store) it on disk. Repeat.
 - **Pipelining**: pass on tuples to parent operations even as an operation is being executed

1/14/2005

Yan Huang - CSCIS330 Database Implementation - Query Processing

Query Processing



Q → Query Plan

Focus: Relational System

- Others?

1/14/2005

Yan Huang - CSCIS330 Database Implementation - Query Processing

Example



Select B,D
From R,S
Where R.A = "c" \wedge S.E = 2 \wedge R.C=S.C

1/14/2005

Yan Huang - CSCIS330 Database Implementation - Query Processing

Relational Algebra - can be used to describe plans...

Ex: Plan I

$$\begin{array}{c}
 \Pi_{B,D} \\
 | \\
 \sigma_{R.A='c' \wedge S.E=2 \wedge R.C=S.C} \\
 | \\
 X \\
 / \quad \backslash \\
 R \quad S
 \end{array}$$

OR: $\Pi_{B,D} [\sigma_{R.A='c' \wedge S.E=2 \wedge R.C=S.C} (RXS)]$

1/14/2005 Yan Huang - CSC5330 Database Implementation - Query Processing

Another idea:

Plan II

$$\begin{array}{c}
 \Pi_{B,D} \\
 | \\
 \bowtie \\
 / \quad \backslash \\
 \sigma_{R.A='c'} \quad \sigma_{S.E=2} \\
 | \quad \quad | \\
 R \quad S
 \end{array}$$

natural join

1/14/2005 Yan Huang - CSC5330 Database Implementation - Query Processing

Example: Estimate costs

$$\begin{array}{c}
 L.Q.P \\
 / \quad | \quad \backslash \\
 P1 \quad P2 \quad \dots \quad Pn \\
 | \quad | \quad \dots \quad | \\
 C1 \quad C2 \quad \dots \quad Cn \\
 \\
 \text{Pick best!} \\
 \uparrow
 \end{array}$$

1/14/2005 Yan Huang - CSC5330 Database Implementation - Query Processing

Relational algebra optimization

- Transformation rules (preserve equivalence)
- What are good transformations?

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: Natural joins & cross products & union

$$R \bowtie S = S \bowtie R$$

$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Note:

- Carry attribute names in results, so order is not important
- Can also write as trees, e.g.:

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: Natural joins & cross products & union

$$R \bowtie S = S \bowtie R$$

$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

$$R \times S = S \times R$$

$$(R \times S) \times T = R \times (S \times T)$$

$$R \cup S = S \cup R$$

$$R \cup (S \cup T) = (R \cup S) \cup T$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: Selects

$$\sigma_{p_1 \wedge p_2}(R) = \sigma_{p_1} [\sigma_{p_2}(R)]$$

$$\sigma_{p_1 \vee p_2}(R) = [\sigma_{p_1}(R)] \cup [\sigma_{p_2}(R)]$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: Project

Let: X = set of attributes
 Y = set of attributes
 XY = X U Y

$$\pi_{xy}(R) = \pi_x[\pi_y(R)]$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: $\sigma + \bowtie$ combined

Let p = predicate with only R attribs
 q = predicate with only S attribs
 m = predicate with only R,S attribs

$$\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S$$

$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: $\sigma + \bowtie$ combined (continued)

Some Rules can be Derived:

$$\sigma_{p \wedge q} (R \bowtie S) =$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) =$$

$$\sigma_{p \vee q} (R \bowtie S) =$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

$$\sigma_{p \wedge q} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) = \sigma_m [(\sigma_p R) \bowtie (\sigma_q S)]$$

$$\sigma_{p \vee q} (R \bowtie S) = [(\sigma_p R) \bowtie S] \cup [R \bowtie (\sigma_q S)]$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: π, σ combined

Let x = subset of R attributes
 z = attributes in predicate P
(subset of R attributes)

$$\pi_x[\sigma_P(R)] = \pi_x \{ \sigma_P [\pi_{xz}(R)] \}$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules: π, \bowtie combined

Let x = subset of R attributes
 y = subset of S attributes
 z = intersection of R, S attributes

$$\pi_{xy}(R \bowtie S) = \pi_{xy} \{ [\pi_{xz}(R)] \bowtie [\pi_{yz}(S)] \}$$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

$\pi_{xy} \{ \sigma_P (R \bowtie S) \} =$

$$\pi_{xy} \{ \sigma_P [\pi_{xz'}(R) \bowtie \pi_{yz'}(S)] \}$$

$z' = z \cup \{ \text{attributes used in } P \}$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules for σ , π combined with X

similar...

e.g., $\sigma_p(R \bowtie S) = ?$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Rules σ , \cup combined:

$\sigma_p(R \cup S) = \sigma_p(R) \cup \sigma_p(S)$

$\sigma_p(R - S) = \sigma_p(R) - S = \sigma_p(R) - \sigma_p(S)$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Which are "good" transformations?

- $\sigma_{p1 \wedge p2}(R) \rightarrow \sigma_{p1}[\sigma_{p2}(R)]$
- $\sigma_p(R \bowtie S) \rightarrow [\sigma_p(R)] \bowtie S$
- $R \bowtie S \rightarrow S \bowtie R$
- $\pi_x[\sigma_p(R)] \rightarrow \pi_x\{\sigma_p[\pi_{xz}(R)]\}$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Conventional wisdom: do projects early

Example: $R(A,B,C,D,E) \quad x=\{E\}$
 $P: (A=3) \wedge (B=\text{"cat"})$

$\pi_x \{ \sigma_P (R) \}$ vs. $\pi_E \{ \sigma_P \{ \pi_{ABE}(R) \} \}$

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

But What if we have A, B indexes?

B = "cat" A=3

Intersect pointers to get pointers to matching tuples

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing

Bottom line:

- No transformation is always good
- Usually good: early selections

1/14/2005 Yan Huang - CSCIS330 Database Implementation - Query Processing
