Database Applications (15-415)

DBMS Internals- Part IX
Lecture 17, March 24, 2014

Mohammad Hammoud
Today...

- **Last Session:**
  - DBMS Internals- Part VIII
    - Algorithms for Relational Operations (Cont’d)

- **Today’s Session:**
  - DBMS Internals- Part IX
    - Query Optimization

- **Announcements:**
  - Project 3 is due on April 5th
  - Final exam is on Sunday, April 27, at 9:00AM in Room 2051 (*all material included*- open book, open notes)
Outline

A Brief Primer on Query Optimization

Query Evaluation Plans

Relational Algebra Equivalences

Estimating Plan Costs

Enumerating Plans
Cost-Based Query Sub-System

Queries

Select * 
From Blah B
Where B.blah = blah

Usually there is a heuristics-based rewriting step before the cost-based steps.

Query Parser

Query Optimizer

Plan Generator

Plan Cost Estimator

Catalog Manager

Schema

Statistics

Query Plan Evaluator
Query Optimization Steps

- **Step 1**: Queries are parsed into internal forms (e.g., parse trees)

- **Step 2**: Internal forms are transformed into ‘canonical forms’ (syntactic query optimization)

- **Step 3**: A *subset* of alternative plans are enumerated

- **Step 4**: Costs for alternative plans are estimated

- **Step 5**: The query evaluation plan with the least estimated cost is picked
The Query Optimizer

- A given query can be evaluated in many ways

- The performance difference between the best and worst ways can be several orders of magnitude

- The query optimizer is responsible for identifying an efficient query plan

- It is unrealistic to expect an optimizer to find the very best plan; it is more important to avoid the worst plans and find a good plan
Outline

A Brief Primer on Query Optimization
Query Evaluation Plans
Relational Algebra Equivalences
Estimating Plan Costs
Enumerating Plans
Query Evaluation Plans

- A query evaluation plan (or simply a plan) consists of an extended relational algebra tree (or simply a tree).

- A plan tree consists of annotations at each node indicating:
  - The access methods to use for each relation
  - The implementation method to use for each operator

- Consider the following SQL query $Q$:

```sql
SELECT S.sname
FROM Reserves R, Sailors S
WHERE R.sid=S.sid AND R.bid=100 AND S.rating>5
```

What is the corresponding RA of $Q$?
Query Evaluation Plans (Cont’d)

- Q can be expressed in relational algebra as follows:

\[ \pi_{sname}(\sigma_{bid=100 \land rating > 5}(\text{Reserves} \bowtie_{sid=sid} \text{Sailors})) \]

A RA Tree:

- \( \bowtie \)
- \( \sigma_{bid=100 \land rating > 5} \)
- \( \bowtie \)
- \( \bowtie \)
- \( \bowtie \)
- \( \bowtie \)

An Extended RA Tree:

- \( \sigma_{bid=100 \land rating > 5} \) (On-the-fly)
- \( \bowtie \) (Simple Nested Loops)
- (File Scan) Reserves
- (File Scan) Sailors
Pipelining vs. Materializing

- When a query is composed of several operators, the result of one operator can sometimes be pipelined to another operator.

Pipeline the output of the join into the selection and projection that follow.
Pipelining vs. Materializing

- When a query is composed of several operators, the result of one operator can sometimes be pipelined to another operator.

Pipeline the output of the join into the selection and projection that follow.

In contrast, a temporary table can be materialized to hold the intermediate result of the join and read back by the selection operation!

Pipelining can significantly save I/O cost!
The I/O Cost of the Q Plan

What is the I/O cost of the following evaluation plan?

\[
\text{Reserves} \quad \text{Sailors}
\]
\[
\text{sid=sid} \quad \text{bid=100} \quad \text{rating > 5}
\]

- Simple Nested Loops

\[
\text{File Scan} \quad \text{(On-the-fly)}
\]

\[
\text{File Scan} \quad \text{(File Scan)}
\]

- The cost of the join is \(1000 + 1000 \times 500 = 501,000\) I/Os (assuming page-oriented Simple NL join)
- The selection and projection are done on-the-fly; hence, do not incur additional I/Os
Pushing Selections

- How can we reduce the cost of a join?
  - By reducing the sizes of the input relations!

```
Reserves
  ▼ sid=sid
    ◊ bid=100
      σ rating > 5
        rating > 5
          ⊓⊔

Sailors
  ▼ sid=sid
    ◊ bid=100
      σ rating > 5
        rating > 5
          ⊓⊔
```

Involves $bid$ in Reserves; hence, “push” ahead of the join!

Involves $rating$ in Sailors; hence, “push” ahead of the join!
Pushing Selections

- How can we reduce the cost of a join?
  - By reducing the sizes of the input relations!
The I/O Cost of the New Q Plan

- What is the I/O cost of the following evaluation plan?

\[
\begin{align*}
\text{Reserves} & \quad \text{Sailors} \\
\quad & \quad \text{sid}=\text{sid} \\
\quad & \quad \text{On-the-fly} \\
\quad & \quad \text{Sort-Merge Join} \\
\quad & \quad \text{Scan; write to temp T1} \\
\quad & \quad \text{Scan; write to temp T2} \\
\end{align*}
\]

Cost of Scanning Reserves = 1000 I/Os
Cost of Writing T1 = 10* I/Os (later)
Cost of Scanning Sailors = 500 I/Os
Cost of Writing T2 = 250* I/Os (later)

*Assuming 100 boats and uniform distribution of reservations across boats.
*Assuming 10 ratings and uniform distribution over ratings.
The I/O Cost of the New Q Plan

- What is the I/O cost of the following evaluation plan?

**Merge Cost** = 10 + 250 = 260 I/Os

**Cost** = 2×2×10 = 40 I/Os
  (assuming B = 5)

**Cost** = 2×4×250 = 2000 I/Os
  (assuming B = 5)
The I/O Cost of the New Q Plan

What is the I/O cost of the following evaluation plan?

- (Scan; write to temp T1) (Reserves)
- (Scan; write to temp T2) (Sailors)
- Sid=sid (Sort-Merge Join)
- Sname (On-the-fly)
- Done on-the-fly, thus, do not incur additional I/Os
The I/O Cost of the New Q Plan

What is the I/O cost of the following evaluation plan?

- **Cost of Scanning Reserves** = 1000 I/Os
- **Cost of Writing T1** = 10 I/Os (later)

- **Cost of Scanning Sailors** = 500 I/Os
- **Cost of Writing T2** = 250 I/Os (later)

- **Merge Cost** = 10 + 250 = 260 I/Os

- **Cost** = 2×2×10 = 40 I/Os (assuming B = 5)

- **Cost** = 2×4×250 = 2000 I/Os (assuming B = 5)

- **Total Cost** = 1000 + 10 + 500 + 250 + 40 + 2000 + 260 = 4060 I/Os

Done on-the-fly, thus, do not incur additional I/Os
The I/O Costs of the Two Q Plans

Total Cost = 501,000 I/Os

Total Cost = 4060 I/Os
Pushing Projections

- How can we reduce the cost of a join?
  - By reducing the sizes of the input relations!

- Consider (again) the following plan:

  - What are the attributes required in the final result?
    - Sid of T1
    - Sid and sname of T2

  Hence, as we scan Reserves and Sailors we can also remove unwanted columns (i.e., “Push” the projections ahead of the join)!
Pushing Projections

- How can we reduce the cost of a join?
  - By reducing the sizes of the input relations!

- Consider (again) the following plan:

```
Scan; write to temp T1
(Restrict Reserves bid=100)
(Produce T1)

Scan; write to temp T2
(Restrict Sailors rating > 5)
(Produce T2)

Cross Join T1 and T2
(Select sid=sid)
(Produce Result)

Select columns from T1: Sid
Select columns from T2: Sid, sname
```

- What are the attributes required from T1 and T2?
  - Sid from T1
  - Sid and sname from T2

Hence, as we scan Reserves and Sailors we can also remove unwanted columns (i.e., “Push” the projections ahead of the join)!
Pushing Projections

- How can we reduce the cost of a join?
  - By reducing the sizes of the input relations!

- Consider (again) the following plan:

```
Reserves (Scan; write to temp T1) [sid=sid]
  ^
  |   sid=sid
  |   ^ name
  |   |   bid=100
  |   |   |   Reserves
  |   |   |   |   (Scan; write to temp T1)
  |   |   |   |   |   Rating > 5
  |   |   |   |   |   |   Sailors (Scan; write to temp T2)
```

"Push" ahead the join

The cost after applying this heuristic can become 2000 I/0s (as opposed to 4060 I/0s with only pushing the selection)!
Using Indexes

- What if indexes are available on Reserves and Sailors?

- With clustered index on `bid` of Reserves, we get 100,000/100 = 1000 tuples (assuming 100 boats and uniform distribution of reservations across boats)

- Since, the index is clustered, the 1000 tuples appear consecutively within the same bucket; thus # of pages = 1000/100 = 10 pages
Using Indexes

- What if indexes are available on Reserves and Sailors?

For each selected Reserves tuple, we can retrieve matching Sailors tuples using the hash index on the sid field.

Selected Reserves tuples need not be materialized and the join result can be pipelined!

For each tuple in the join result, we apply rating > 5 and the projection of sname on-the-fly.
Using Indexes

- What if indexes are available on Reserves and Sailors?

Is it necessary to project out unwanted columns? NO, since selection results are NOT materialized

(Clustered hash index on bid) Reserves

(Hash index on sid) Sailors

(Use hash index; do not write result to temp) bid=100

(On-the-fly) sid=sid

(On-the-fly) rating > 5

(On-the-fly) sname

(Hash index on sid) bid=100
What if indexes are available on Reserves and Sailors?

- Hash index on sid

Does the hash index on sid need to be clustered?

NO, since there is at most 1 matching Sailors tuple per a Reserves tuple! Why?
What if indexes are available on Reserves and Sailors?

1. Hash index on sid
   - Cost = 1.2 I/Os (if A(1)) or 2.2 (if A(2))

2. (Clustered hash index on bid)

3. Index Nested Loops, with pipelining
   - (Hash index on sid)

4. (On-the-fly)
   - sname
   - rating > 5
Using Indexes

- What if indexes are available on Reserves and Sailors?

Why not *pushing* this selection ahead of the join?

It would require a scan on Sailors!
What is the I/O cost of the following evaluation plan?

\[\text{Total Cost} = 10 + 1200 = 1210 \text{ I/Os} \]
Comparing I/O Costs: Recap

\[ \text{Total Cost} = 501,000 \text{ I/Os} \]

\[ \text{Total Cost} = 4060 \text{ I/Os} \]
But, How Can we Ensure Correctness?

Canonical form

Still the same result!

How can this be guaranteed?
Outline

- A Brief Primer on Query Optimization
- Query Evaluation Plans
- Relational Algebra Equivalences
- Estimating Plan Costs
- Enumerating Plans
Relational Algebra Equivalences

- A relational query optimizer uses *relational algebra equivalences* to identify many *equivalent* expressions for a given query.

- Two relational algebra expressions over the same set of input relations are said to be *equivalent* if they produce the same result on all relations’ instances.

- Relational algebra equivalences allow us to:
  - Push selections and projections ahead of joins
  - Combine selections and cross-products into joins
  - Choose different join orders
RA Equivalences: Selections

- Two important equivalences involve selections:
  1. Cascading of Selections:
     \[ \sigma_{c_1 \land \ldots \land c_n}(R) \equiv \sigma_{c_1} \left( \ldots \sigma_{c_n}(R) \right) \]
     Allows us to combine several selections into one selection
     OR: Allows us to replace a selection with several smaller selections
  2. Commutation of Selections:
     \[ \sigma_{c_1} \left( \sigma_{c_2}(R) \right) \equiv \sigma_{c_2} \left( \sigma_{c_1}(R) \right) \]
     Allows us to test selection conditions in either order
RA Equivalences: Projections

- One important equivalence involves projections:
  - Cascading of Projections:

$$\pi_{a_1}(R) \equiv \pi_{a_1}(\ldots(\pi_{a_n}(R)))$$

This says that successively eliminating columns from a relation is equivalent to simply eliminating all but the columns retained by the final projection!
RA Equivalences: Cross-Products and Joins

- Two important equivalences involve cross-products and joins:

  1. **Commutative Operations:**

     
     \[(R \times S) \equiv (S \times R)\]
     \[(R \bowtie S) \equiv (S \bowtie R)\]

     This allows us to choose which relation to be the inner and which to be the outer!
RA Equivalences: Cross-Products and Joins

- Two important equivalences involve cross-products and joins:

  2. **Associative Operations:**

  $R \times (S \times T) \equiv (R \times S) \times T$
  
  $R \bowtie (S \bowtie T) \equiv (R\bowtie S) \bowtie T$

  It follows: $R \bowtie (S \bowtie T) \equiv (T\bowtie R) \bowtie S$

This says that regardless of the order in which the relations are considered, the final result is the same!

This *order-independence* is fundamental to how a query optimizer generates alternative query evaluation plans.
RA Equivalences: Selections, Projections, Cross Products and Joins

- Selections with Projections:

\[ \pi_a (\sigma_c (R)) \equiv \sigma_c (\pi_a (R)) \]

This says we can commute a selection with a projection if the selection involves only attributes retained by the projection!

- Selections with Cross-Products:

\[ R \bowtie_c T \equiv \sigma_c (R \times S) \]

This says we can combine a selection with a cross-product to form a join (as per the definition of a join)!
RA Equivalences: Selections, Projections, Cross Products and Joins

- Selections with Cross-Products and with Joins:

\[ \sigma_c(R \times S) \equiv \sigma_c(R) \times S \]
\[ \sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S \]

**Caveat:** The attributes mentioned in \( c \) must appear only in \( R \) and **NOT** in \( S \)

This says we can commute a selection with a cross-product or a join if the selection condition involves only attributes of one of the arguments to the cross-product or join!
RA Equivalences: Selections, Projections, Cross Products and Joins

- Selections with Cross-Products and with Joins (Cont’d):

\[ \sigma_c(R \times S) \equiv \sigma_{c_1 \land c_2 \land c_3}(R \times S) \]

\[ \equiv \sigma_{c_1}(\sigma_{c_2}(\sigma_{c_3}(R \times S))) \]

\[ \equiv \sigma_{c_1}(R \times \sigma_{c_3}(S)) \]

This says we can push part of the selection condition \( c \) ahead of the cross-product!

This applies to joins as well!
RA Equivalences: Selections, Projections, Cross Products and Joins

- Projections with Cross-Products and with Joins:

\[
\pi_a(R \times S) \equiv \pi_{a_1}(R) \times \pi_{a_2}(S)
\]

\[
\pi_a(R \bowtie_c S) \equiv \pi_{a_1}(R) \bowtie c \pi_{a_2}(S)
\]

\[
\pi_a(R \bowtie c S) \equiv \pi_a(\pi_{a_1}(R) \bowtie c \pi_{a_2}(S))
\]

Intuitively, we need to retain only those attributes of R and S that are either mentioned in the join condition \( c \) or included in the set of attributes \( a \) retained by the projection.
How to Estimate the Cost of Plans?

- Now that correctness is ensured, how can the DBMS estimate the costs of various plans?

![Diagram of SQL queries and their canonical form](image-url)
Next Class

Queries

Query Optimization and Execution

Relational Operators

Files and Access Methods

Buffer Management

Disk Space Management

Transaction Manager

Lock Manager

Recovery Manager

DB

Continue...