Relational Query Optimization

Yanlei Diao

Slide Content Courtesy of R. Ramakrishnan, J. Gehrke, and J. Hellerstein
Overview of Query Evaluation

- **Query evaluation plan**: tree of *relational algebra* operators, with choice of algorithm for each operator.

- **Query optimization**: given a query, many plans are possible
  - Ideally, find the most efficient plan.
  - In practice, avoid worst plans in practice.
SQL Refresher

SELECT {DISTINCT} <list of columns>
FROM <list of relations>
{WHERE <list of "Boolean Factors">}
{GROUP BY <list of columns>}
{HAVING <list of Boolean Factors>}
{ORDER BY <list of columns>};

- Query Semantics:
  1. Take Cartesian product (a.k.a. cross-product) of relns in FROM, projecting only to those columns that appear in other clauses
  2. If a WHERE clause exists, apply all filters in it
  3. If a GROUP BY clause exists, form groups on the result
  4. If a HAVING clause exists, filter groups with it
  5. If an ORDER BY clause exists, make sure output is in right order
  6. If there is a DISTINCT modifier, remove duplicates
Basics of Query Optimization

- Convert selection conditions to *conjunctive normal form* (CNF):
  - `(day<8/9/94 OR bid=5 OR sid=3) AND (rname= ‘Paul’ OR sid=3)`

- Interleave FROM and WHERE into an *operator tree for optimization*.
  - Query optimization largely works for Conjunctive Queries (only).

- Apply GROUP BY, HAVING, DISTINCT and ORDER BY at the end, pretty much in that order.
Outline of topics

- Query plans and equivalences

- Query optimization issues
  - Plan space
  - Cost estimation
  - Plan search

- Handling nested queries

- Multi-objective optimization in Cloud Computing
SELECT  S.sname
FROM    Reserves R, Sailors S
WHERE  R.sid=S.sid AND
        R.bid=100 AND S.rating>5

Expression in Relational Algebra (RA):

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

- **Query evaluation plan** extends an RA tree with:
  1. *access method* for each relation;
  2. *implementation method* for each other operator.

- What are the missed opportunities?
  - Selections could have been `pushed’ earlier.
  - Use of indexes.
  - More efficient joins.
Relational Algebra Equivalences

- **Selections:** \( \sigma_{c_1 \land \ldots \land c_n}(R) \equiv \sigma_{c_1}(\ldots \sigma_{c_n}(R)) \) (Cascade)

- **Projections:** \( \pi_{a_1}(R) \equiv \pi_{a_1}(\ldots(\pi_{a_1,\ldots,a_n}(R))) \) (Cascade)

- **Joins:** \( (R \bowtie S) \equiv (S \bowtie R) \) (Commutative)

\[ R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T \] (Associative)

Show that: \( R \bowtie (S \bowtie T) \equiv (T \bowtie R) \bowtie S \)
More Equivalences

- $\sigma_c (R \times S) \equiv R \bowtie_c S$

- $\sigma_c (R \bowtie S) \equiv \sigma_c (R) \bowtie S$, if $c$ is only applied to $R$

- $\pi_a (\sigma_c (R)) \equiv \sigma_c (\pi_a (R))$ holds if $\sigma$ only uses attributes retained by $\pi$

- For $\pi_b (R \bowtie_a S)$, we can ‘push’ $\pi$ before $\bowtie$ by retaining both the $a$ attribute and the $b$ attribute (if existent)

☞ But, aggregates do not commute with other operators.
Schema for Examples

Sailors \((\text{sid}: \text{integer}, \text{sname}: \text{string}, \text{rating}: \text{integer}, \text{age}: \text{real})\)
Reserves \((\text{sid}: \text{integer}, \text{bid}: \text{integer}, \text{day}: \text{dates}, \text{rname}: \text{string})\)

- Reserves:
  - Each tuple is 40 bytes long, 100 tuples per page, 1000 pages.
- Sailors:
  - Each tuple is 50 bytes long, 80 tuples per page, 500 pages.
Query Plan 1 (Selection Pushed Down)

- **Push selections below the join.**

- **Materialization vs. Pipelining:**
  - **Materialize** a temporary relation T, if the next operator needs to scan T multiple times.
  - **Pipelining:** the opposite.

- **With 5 buffer pages, cost of plan:**
  - Scan Reserves (1000) + write temp T1 (10 pages, if we have 100 boats, uniform distribution).
  - Scan Sailors (500) + write temp T2 (250 pages, if we have 10 ratings).
  - **Sort-Merge join:** Sort T1 (2*2*10), sort T2 (2*4*250), merge (10+250).
  - Total = 4060 page I/Os.
Query Plan 2 (Different Join Method)

- Change the join method to *block nested loops join*.

With 5 buffer pages, cost of plan:
- Scan Reserves (1000) + write temp T1 (10 pages).
- Scan Sailors (500) + write temp T2 (250 pages).
- **BNL join**: join cost = 10 + 4 * 250.
- Total cost = 2770.
Indexes

- A tree index *matches* (a conjunction of) terms if the attributes in the terms form a *prefix* of the search key.
  
  - Tree index on `<a, b, c>`
  - `a=5 AND b=3` ?
  - `a=5 AND b>6` ?
  - `b=3` ?
Query Plan 3 (Using Indexes)

- **Selection using index**: clustered index on bid of Reserves.
  - Retrieve 100,000/100 = 1000 tuples
  - Clustering: read 1000/100 = 10 pages.

- Indexed NLJ: pipeline the outer and index lookup on sid of Sailors.
  - The outer: no need to materialize.
  - The inner: sid is a key; at most one match tuple, unclustered index OK.

- **Cost**: (rough illustration, need more info. for precise calculation)
  - Selection of Reserves tuples (~10 I/Os).
  - For each tuple, get matching Sailor tuple (1000*(2~3)).
  - Total = 2010~3010 I/Os.