Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 - DB Applications

C. Faloutsos – A. Pavlo
Lecture#14(b): Implementation of
Relational Operations

Administrivia

• HW4 is due today.
• HW5 is out.

• Mid-term on Tues March 4th
  – Will cover everything up to and including this
    week’s lectures.
  – Closed book, one sheet of notes (double-sided).
• Please email Christos+Andy if you need
  special accommodations.
• See exam guideline:
Extended Office Hours

- **Christos:**
  - Friday Feb 28th 3:00pm-5:00pm
- **Andy:**
  - Friday Feb 28th 10:00am-12:00pm
  - Monday Mar 3rd 9:00am-10:00am
  - Tuesday Mar 4th 10:00am-12:00pm

Last Class: Selections

- **Approach #1:** Find the cheapest access path, retrieve tuples using it, and apply any remaining terms that don’t match the index
- **Approach #2:** Use multiple indexes to find the intersection of matching tuples.

Last Class: Joins

- Nested Loop Joins
- Index Nested Loop Joins
- Sort-Merge Joins
- Hash Joins
Today’s Class

• Set Operations
• Aggregate Operations
• Explain
• Mid-term Review + Q&A

Set Operations

• Intersection ($R \cap S$)
• Cross-Product ($R \times S$)
• Union ($R \cup S$)
• Difference ($R - S$)

Special case of join. Use same techniques from last class.

We can use sorting or hashing strategies.

Union/Difference – Sorting

• Sort both relations on combination of all attributes.
• Scan sorted relations and merge them.
  – For union, just eliminate duplicates as we go.
  – For difference, we emit tuples from $R$ if they don’t appear in $S$. 
Union/Difference

- Sort both relations on combination of **all** attributes.
- Scan sorted relations and merge them.
  - For union, just eliminate duplicates as we go.
  - For difference, we emit tuples from R if they don’t appear in S.

Union/Difference – Hashing

- Partition R and S using hash function $h_1$.
- For each S-partition, build in-memory hash table (using $h_2$), scan corresponding R-partition and add tuples to table.
  - For union, discard duplicates.
  - For difference, probe the hash table for S and emit R tuples that are missing.
Today's Class

* Set Operations
* Aggregate Operations
* Explain
* Mid-term Review + Q&A

Aggregate Operators

* Basic SQL-92 aggregate functions:
  - MIN – Return the minimum value.
  - MAX – Return the maximum value.
  - SUM – Return the sum.
  - COUNT – Return a count of the # of rows.
  - AVG – Return the average value.
* Note that each can be executed with or without GROUP BY.
Running Totals

```sql
SELECT bid, COUNT(*)
FROM Reserves
GROUP BY bid;
```

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
<th>rname</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>102</td>
<td>2014-02-01</td>
<td>matlock</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>2014-02-02</td>
<td>macgyver</td>
</tr>
<tr>
<td>11</td>
<td>101</td>
<td>2014-02-01</td>
<td>dallas</td>
</tr>
</tbody>
</table>

Aggregate Operators

Without grouping:
- In general, requires scanning the relation.
- Given index whose search key includes all attributes in the `SELECT` or `WHERE` clauses, can do index-only scan.

With grouping, we have three approaches:
- Sorting
- Hashing
- A suitable tree-based index
Aggregates with Grouping

• **Approach #1: Sorting**
  – Sort on group-by attributes, then scan relation and compute aggregate for each group.

• **Approach #2: Hashing**
  – Build in-memory hash table on group-by attributes. Update running totals for each tuple that we examine.

Aggregates with Grouping

• **Approach #3: Indexes**
  – Given tree index whose search key includes all attributes in `SELECT`, `WHERE`, and `GROUP BY` clauses, we can do index-only scan.
  – If `GROUP BY` attributes form prefix of search key, we can retrieve data entries/tuples in group-by order.

Today’s Class

• Set Operations
• Aggregate Operations
• Explain
• Mid-term Review + Q&A
When you precede a `SELECT` statement with the keyword `EXPLAIN`, the DBMS displays information from the optimizer about the statement execution plan.

The system “explains” how it would process the query, including how tables are joined and in which order.

**Pseudo Query Plan:**

```
SELECT bid, COUNT(*) AS cnt
FROM Reserves
GROUP BY bid
ORDER BY cnt
```

**Sort Plan:**

```
<table>
<thead>
<tr>
<th>Operation</th>
<th>Rows</th>
<th>Est. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort</td>
<td>15000</td>
<td>49.24 ms</td>
</tr>
<tr>
<td>Reserves</td>
<td>15000</td>
<td>84.88 ms</td>
</tr>
</tbody>
</table>
```

This plan shows the estimated number of rows and the estimated time for each operation.
EXPLAIN

```sql
EXPLAIN SELECT bid, COUNT(*) AS cnt
FROM Reserves
GROUP BY bid
ORDER BY cnt
```

MySQL v5.5

EXPLAIN ANALYZE

- **ANALYZE** option causes the statement to be actually executed.
- The actual runtime statistics are displayed
- This is useful for seeing whether the planner's estimates are close to reality.
- Note that **ANALYZE** is a Postgres idiom.

EXPLAIN ANALYZE

```sql
EXPLAIN ANALYZE
SELECT bid, COUNT(*) AS cnt
FROM Reserves
GROUP BY bid
ORDER BY cnt
```

Postgres v9.1
EXPLAIN ANALYZE

• Works on any type of query.
• Since **ANALYZE** actually executes a query, if you use it with a query that modifies the table, that modification **will** be made.

Today’s Class

• Set Operations
• Aggregate Operations
• Explain
• Mid-term Review + Q&A

Mid-term Review

• Everything from the beginning of the course to today is fair game:
  – 01intro.pdf to 14RelOp.pdf
• Bring a calculator. Your phone is unfortunately not a calculator.
Relational Model

- Chapters 2-4
- E-R Diagrams
- Relational Algebra
- Relational Calculus

SQL

- Chapter 5
- Basic Syntax
- Different Types of Joins
- Nested Queries

Storage & Indexes

- Chapters 8-10
- How a DBMS stores data on disk.
- B-Tree Indexes
- Hash Table Indexes
- Make sure you know costs + trade-offs.
Query Evaluation

- Chapters 12-14
- Sorting
- Hashing
- Selection + Access Paths
- Join Algorithms

Questions?