CMU SCS

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

C. Faloutsos – A. Pavlo

Lecture#14(b): Implementation of Relational Operations

CMU SCS

Administrivia

- HW4 is due today.
- HW5 is out.

```
MongoDB shell version: 2.4.9
connecting to: 15-415
> db.homeworks.findOne({'assignment': 'HW5'}, {'duedate': 1})
{
        "_id" : ObjectId("530ea368bd49f60289584ff4"),
        "duedate" : "2014-03-18 13:30"
}
```


CMU SCS

Administrivia

- 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:
  - http://bit.ly/1hUPeGV

CMU SCS

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

CMU SCS

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.

CMU SCS

Last Class: Joins

- Nested Loop Joins
- Index Nested Loop Joins
- Sort-Merge Joins
- Hash Joins

CMU SCS

Today's Class

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

CMU SCS

Set Operations

- Intersection (**R∩S**)
- Cross-Product (**R×S**)

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

- Union (**R∩S**)
- Difference (**R-S**)

We can use sorting or hashing strategies.

CMU SCS

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.

CMU SCS

Union/Difference

External Merge Sort
from Lecture #12

- 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.

Faloutsos/Pavlo CMU SCS 15-415/615 10

CMU SCS

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.

Faloutsos/Pavlo CMU SCS 15-415/615 11

CMU SCS

Union/Difference – Hashing

Two-Phase Hashing
from Lecture #13

- Partition R and S using hash
- 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.

Faloutsos/Pavlo CMU SCS 15-415/615 12

CMU SCS

Today's Class

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

CMU SCS

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**.

CMU SCS

Aggregate Operators

- Scan the relation and maintain running information about matched tuples.
  - **MIN** – Smallest value seen thus far.
  - **MAX** – Largest value seen thus far.
  - **SUM** – Total of the values seen thus far.
  - **COUNT** – The # of rows seen thus far.
  - **AVG** – **<Total,Count>** of the values seen.

CMU SCS

Running Totals

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

sid	bid	day	rname
6	103	2014-02-01	matlock
1	102	2014-02-02	macgyver
2	101	2014-02-02	a-team
1	101	2014-02-01	dallas

bid	count
6	2
1	1
2	1

Hash Table

CMU SCS

Aggregate Operators

```
SELECT COUNT(*) FROM Reserves
```

- 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.

CMU SCS

Aggregate Operators

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

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

CMU SCS

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.

Faloutsos/Pavlo CMU SCS 15-415/615 19

CMU SCS

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.

Faloutsos/Pavlo CMU SCS 15-415/615 20

CMU SCS

Today's Class

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

Faloutsos/Pavlo CMU SCS 15-415/615 21

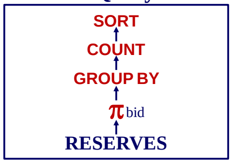
EXPLAIN

- 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.

EXPLAIN

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

Pseudo Query Plan:



EXPLAIN

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

```
15-415=# EXPLAIN SELECT bid, COUNT(*) AS cnt FROM reserves GROUP BY bid ORDER BY cnt;
                              QUERY PLAN
------------------------------------------------------------------------
 Sort  (cost=48.74..49.24 rows=200 width=4)
   Sort Key: (count(*))
   -> HashAggregate  (cost=39.10..41.10 rows=200 width=4)
        -> Seq Scan on reserves  (cost=0.00..29.40 rows=1940 width=4)
(4 rows)
```

Postgres v9.1

CMU SCS

EXPLAIN

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

```
mysql> EXPLAIN SELECT bid, COUNT(*) AS cnt FROM reserves GROUP BY bid ORDER BY cnt;
+----+-------------+----------+-------+---------------+------+---------+------+------+----------------------------------------------+
| id | select_type | table    | type  | possible_keys | key  | key_len | ref  | rows | Extra                                        |
+----+-------------+----------+-------+---------------+------+---------+------+------+----------------------------------------------+
|  1 | SIMPLE      | reserves | index | NULL          | bid  | 4       | NULL |   10 | Using index; Using temporary; Using filesort |
+----+-------------+----------+-------+---------------+------+---------+------+------+----------------------------------------------+
1 row in set (0.00 sec)
```

MySQL v5.5

CMU SCS

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.

CMU SCS

EXPLAIN ANALYZE

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

```
15-415=# EXPLAIN ANALYZE SELECT bid, COUNT(*) AS cnt FROM reserves GROUP BY bid ORDER BY cnt;
                                                  QUERY PLAN
---------------------------------------------------------------------------------------------------------
 Sort  (cost=48.74..49.24 rows=200 width=4) (actual time=0.020..0.020 rows=4 loops=1)
   Sort Key: (count(*))
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=39.10..[illegible] rows=200 width=[illegible]) (actual time=0.013..0.014 rows=4 loops=[illegible])
         ->  Seq Scan on reserves  (cost=0.00..29.40 rows=[illegible] width=4) (actual time=0.006..0.00[illegible] rows=10 loops=1)
 Total runtime: 0.051 ms
(6 rows)
```

Postgres v9.1

CMU SCS

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.

Faloutsos/Pavlo CMU SCS 15-415/615 28

CMU SCS

Today's Class

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

Faloutsos/Pavlo CMU SCS 15-415/615 29

CMU SCS

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.

Faloutsos/Pavlo CMU SCS 15-415/615 30

CMU SCS

Relational Model

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

CMU SCS

SQL

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

CMU SCS

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.

CMU SCS

Query Evaluation

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

CMU SCS

Questions?