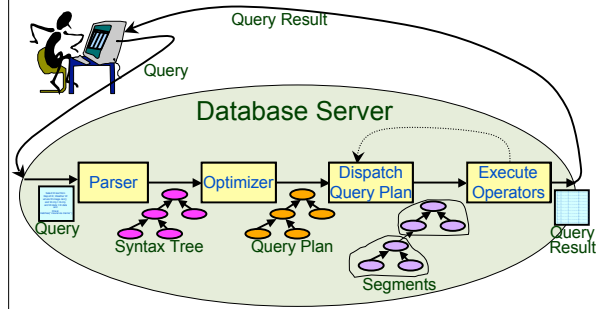


Evaluation of Relational Operations

Chapter 12 and 14

Query Execution Life-Cycle

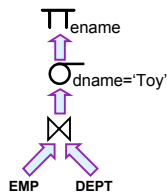


Query Evaluation Plan

- Extended relational algebra tree, including algorithms for each operator

EMP (ssn, ename, addr, sal, did)
DEPT (did, dname, floor, mgr)

```
SELECT E.ename  
FROM Emp E, Dept D  
WHERE D.dname = 'Toy'  
AND D.did = E.did
```



Query Optimizer selects the evaluation plan

Operator Evaluation

- How to implement common operators?
 - Selection
 - Projection (optional DISTINCT)
 - Join
 - Set Difference
 - Union
 - Aggregate operators (SUM, MIN, MAX, AVG)
 - GROUP BY
- Next week – How to choose a plan?
- Catalogs consulted to parse, optimize, execute plan

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

4

System Catalogs

- For each relation:
 - name, file name, file structure (e.g., Heap file)
 - attribute name and type, for each attribute
 - index name, for each index
 - integrity constraints
- For each index:
 - structure (e.g., B+ tree) and search key fields
- For each view:
 - view name and definition
- Plus statistics, authorization, buffer pool size, etc.

Catalogs are themselves stored as relations!

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

5

Attribute Catalog

attrName	relName	type	position
attr_name	Attribute_Cat	string	1
rel_name	Attribute_Cat	string	2
type	Attribute_Cat	string	3
position	Attribute_Cat	integer	4
sid	Students	string	1
name	Students	string	2
login	Students	string	3
age	Students	integer	4
gpa	Students	real	5
fid	Faculty	string	1
fname	Faculty	string	2
sal	Faculty	real	3

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

6

Selection

- An **access path** is a method of retrieving tuples and applying some (potentially empty) selection condition
- Example of a selection condition
 - a predicate: $\text{gpa} > 3.0$ and $\text{age} = 21$
- Examples of access paths
 - File scan
 - Index that *matches* a selection in the query. Examples:
 - B+tree index on the $\langle \text{gpa}, \text{age} \rangle$ attributes
 - B+tree index on gpa
 - B+tree index on age
 - Hash index on age

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 7

Selection

- Where $R.a \text{ op value}$
- Options:

■ Heap file	Cost: $O(N)$
■ Sorted File	Cost: $O(\log_2 N) + \dots$
■ Index	
■ Hash	Cost: $O(1) + \dots$
■ B+Tree: Clustered/Unclustered	Cost: $O(\log_p N) + \dots$

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 8

Index Matching

- When can we use an index to evaluate a selection **predicate**?
- Convert to Conjunctive Normal Form (CNF) **(p1 or p3) and (p2 or p3)**
- An index *matches* a predicate if index can be used to evaluate the predicate
- An index can match subset of conjuncts
 - *Primary conjuncts*

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 9

Index Matching

- Search key <a, b, c>

Tree Idx	Hash Idx
• yes	• no!
• yes	• no!
• no!	• no!
• yes	• yes
• yes	• no!

 - a=5 and b= 3?
 - a > 5 and b < 3
 - b=3
 - a=7 and b=5 and c=4 and **d>4**
 - a=7 and c=5
- Index matches (part of) a predicate if
 - Conjunction of terms involving only attributes (no disjunctions)
 - Hash: only equality operation, predicate has all index attributes.
 - Tree: Attributes are a prefix of the search key, any ops.

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 10

Index Matching

- Predicate could match more than 1 index
- Hash index on <a, b> and B+tree index on <a, c>
 - a=7 and b=5 and c=4 Which index?
 - Option1: Use either (or a file scan!)
 - Check selectivity of primary conjunct
 - Option2: Use both! Algorithm: Intersect rid sets.
 - Sort rids, retrieve rids in both sets.

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 11

Selection

- Hash index on <a> and Hash index on
 - a=7 **or** b>5 Which index?
 - Neither! File scan required for b>5
- Hash index on <a> and B+-tree on
 - a=7 **or** b>5 Which index?
 - Option 1: Neither
 - Option 2: Use both! Fetch rids and union
 - Look at selectivities closely. Optimizer!
- Hash index on <a> and B+-tree on
 - (a=7 **or** c>5) and b > 5 Which index?
 - Could use B+-tree (check selectivity)

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 12

When to use a B+tree index

- Consider
 - A relation with 1M tuples
 - 100 tuples on a page
 - 500 (key, rid) pairs on a page

data pages = $1M/100 = 10K$ pages
 # leaf idx pgs = $1M / (500 * 0.67) \sim 3K$ pages

	1% Selection	10% Selection
Clustered	30 + 100	300 + 1000
Non-Clustered	30 + 10,000	300 + 100,000
NC + Sort Rids	30 + ($\sim 10,000$)	300 + ($\sim 10,000$)

⇒ Choice of Index access plan, consider:
 1. Index Selectivity 2. Clustering
 ⇒ Similar consideration for hash-based indices

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 13

Projection

- Select DISTINCT R.a, R.d
 - Remove attributes
 - Eliminate duplicates
- Algorithms
 - Sorting: Sort on all the projected attributes
 - Pass 0: eliminate unwanted fields. Tuples in the sorted-runs may be smaller
 - Eliminate duplicates in the merge pass & in-memory sort
 - Hashing: Two phases
 - Partitioning
 - Duplicate elimination

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 14

Hashing

Can $h_1 = h_2$?

What if the hash table for a partition overflows, i.e. can't fit in memory?

Recursively apply hash-based projection technique to handle partition overflow problem

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 15

Projection

- Sort-based approach
 - better handling of skew
 - result is sorted
 - I/O costs are comparable if $B^2 > |R|$
- Index-only scan
 - Projection attributes subset of index attributes
 - Apply projection techniques to data entries (much smaller!)
- If an ordered (i.e., tree) index contains all projection attributes as *prefix* of search key:
 - Retrieve index data entries in order
 - Discard unwanted fields
 - Compare adjacent entries to eliminate duplicates (if required)

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 16

Join Operator

```
SELECT *
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
```


- Commercial systems spend a lot of effort optimizing equality joins
 - Why is this important?
 - What is the major source of performance cost when joining two (large) relations?
- Cost Metric: # of I/Os
 - (We'll ignore output cost)

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 17

Join Operator

- Many different ways of evaluating joins

Sailors		Reserves	
sid	name	sid	bid
1	Lucky	1	100
2	Rusty	1	200
3	Bob	3	300
4	Fred	4	200



How would you evaluate this join in memory?

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 18

Simple Nested Loops Join

```
foreach tuple r in R do
  foreach tuple s in S do
    if r.sid == s.sid then add <r, s> to result
```

- Cost Model

- $||R||$ = # tuples in R
- $|R|$ = # pages in R

Slightly different notation from textbook!

- How many I/Os ?

$$|R| + ||R|| * |S|$$

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

19

Page-Oriented Nested Loops

- Page-oriented Nested Loops join:

- For each page of R, get each page of S, and join
- How many I/Os?
 $|R| + |R| * |S|$
- If S is the outer, then $|S| + |R| * |S|$

How many buffer pages does this use?

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

20

Block Nested Loops

- Can we exploit available memory?
- Suppose we have B buffer pages available.
- Use B-2 pages to hold a block of **outer** R.

```
foreach block of B-2 pages of R do
  foreach page of S do
    foreach r in the B-2 R pages do
      foreach tuple s in the S page do
        if r.sid == s.sid then add <r, s> to result
```

- Cost: $|R| + |S| * \left\lceil \frac{|R|}{B-2} \right\rceil$

What is the cost if the smaller relation fits entirely in memory?

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

21

Joining in-memory tuples

Small overhead for the hash table, but big savings in CPU costs

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 22

Index Nested Loops

```

foreach tuple r in R do
  Probe Index on S.sid
  foreach matching tuple s do add <r, s> to result
  
```

- Use index on join attribute of the inner relation
 - Cost: $|R| + (|R| * \text{cost of finding matching S tuples})$
- Index cost:
 - 1-2 for hash index
 - 2-4 for B+ tree.
 - Clustered index: 1 I/O (typical), unclustered: up to 1 I/O per matching S tuple.

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 23

Sort-Merge

- Sort R on the join attribute (if necessary)
- Sort S on the join attribute (if necessary)
- Merge

How many buffer pages are used in the merge?

Sorted R

Sorted S

- Cost: $\sim |R| \log |R| + |S| \log |S| + (|R|+|S|)$
- Worst Case: $\sim |R| \log |R| + |S| \log |S| + (|R|*|S|)$
 - When?
- Backups needed if #duplicates in S exceeds buffer size

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 24

Grace Hash Join

- Step 1: (Build) Hash both relations, R and S, on the join attribute, producing k disk-based partitions
 - Guarantees that R tuples can only join with S tuples in the same partition
- Step 2: (Probe) Read in complete partition of (smaller relation) R, and scan S partition for matches

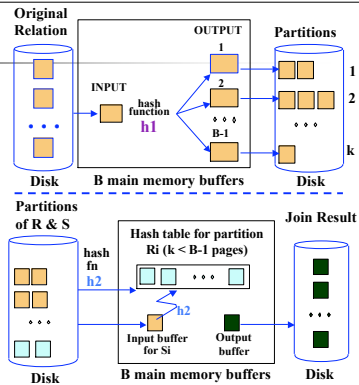
3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

25

Hash Join

- $h_1 \neq h_2$



3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

26

Hash Join

- Partition phase, read+write both relations; $2(|R|+|S|)$.
- In probe phase, read both relations; $|R|+|S|$ I/Os.
 - Assumes each R partition fits in memory in probe phase
- This is *Grace hash join*
 - Variant hybrid hash join, saves more I/Os
 - Combine partition and probe of first partition

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

27

Join Algorithms

- Hash-Join vs. Sort-Merge Cost:
 - With minimum memory, each costs $3(|R|+|S|)$ I/Os
 - Memory requirements (rough):
 - SM: Larger relation $|S| \leq B^2$
 - Hash: Smaller relation $|R| \leq B^2$
 - Hash Join superior if relation sizes differ greatly.
 - Hash Join is highly parallelizable.
 - SM less sensitive to data skew; result is sorted.
- When R and S fit in memory:
Hash-Join = Block NL with hashing

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 28

General Join Conditions

- Equalities over several attributes
e.g., $R.sid=S.sid$ AND $R.rname=S.rname$:
 - Index NL
 - index on $\langle sid, rname \rangle$
 - index on sid or $rname$.
 - SM and Hash, sort/hash on combination of join attrs
- Inequality conditions (e.g., $R.rname < S.rname$):
 - For Index NL, need (clustered!) B+ tree index.
 - Large # index matches
 - SM and Hash not applicable
 - Block NL likely to be the winner

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 29

Set Operations

- \cap and \times special cases of join
- \cup and $-$ similar; we'll do \cup .
 - Duplicate elimination
- Sorting:
 - Sort both relations (on all attributes).
 - Merge sorted relations eliminating duplicates.
 - *Alternative:* Merge sorted runs from both relations.
- Hashing:
 - Partition R and S
 - Build hash table for R_i .
 - Probe with tuples in S_i , add to table if not a duplicate

3/10/09 EECS 484: Database Management Systems, Kristen LeFevre 30

Aggregates

- Sorting
 - Sort on group by attributes (if any)
 - Scan sorted tuples, computing running aggregate
 - Max: Max
 - Average: Sum, Count
 - If the group by attribute changes, output aggregate result
- Cost: sorting cost

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

31

Aggregates

- Hashing
 - Hash on group by attributes (if any)
 - Hash entry: group attributes + running aggregate
 - Scan tuples, probe hash table, update hash entry
 - Scan hash table, and output each hash entry
- Cost: Scan relation!
- What if we have a large # groups?

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

32

Aggregates

- Index
 - Without Grouping
 - Can use B+tree on aggregate attribute(s)
 - Where clause?
 - With grouping
 - B+tree on all attributes in SELECT, WHERE and GROUP BY clauses
 - Index-only scan
 - If group-by attributes prefix of search key => data entries/tuples retrieved in group-by order
 - Else => get data entries and then use a sort or hash aggregate algorithm

3/10/09

EECS 484: Database Management Systems, Kristen LeFevre

33



Announcements

- Optional Exercises:
 - 12.1 (1-4), 12.3, 12.5
 - 13.1, 13.3
 - 14.1 (2, 3, 4, 6, 7, 8, 9, 10), 14
