CS143:Evaluation and Optimization

Ch 13-14

1

Basic Steps in Query Processing

1. Parsing and translation:
   ... Translation of SQL queries into naïve RA
2. Optimization: of RA expressions by
   1. Pushing selection and projection
   2. Estimating cost of different join orders, and selecting that of minimum cost
   3. Selecting best implementation for each operator.
3. Evaluation

2

Join Operation

- Several different algorithms to implement joins
  - Nested-loop join
  - Block nested-loop join
  - Indexed nested-loop join
  - Merge-join
  - Hash-join
- Choice based on cost estimate

3

R JOIN S?

Nested Join

R

C	
40	T1
60	T2
30	T3
10	T4
20	T5

S

C	
10	T6
60	T7
40	T8
20	T9

4

- Nested-Loop Join

  For each r ? R do

  For each s ? S do

  if r.C = s.C then output r,s pair

R

40	T1
60	T2
30	T3
10	T4
20	T5

S

10	T6
60	T7
40	T8
20	T9

R is the outer relation and S is the inner relation

5

Block Nested-Loop Join

- Variant of nested-loop join in which every block of inner relation is paired with every block of outer relation.

```
for each block B_r of r do begin
    for each block B_s of s do begin
        for each tuple t_r in B_r do begin
            for each tuple t_s in B_s do begin
                Check if (t_r,t_s) satisfy the join condition
                if they do, add t_r • t_s to the result.
            end
        end
    end
end
```

6

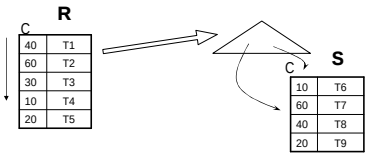
- Index Join (conceptual)

(1) If index supporting S.C does not exist , create one, and

(2) For each r ? R do

X ? index-lookup(S.C, r.C)

For each s ? X, output (r,s)



7

Selection and Indexing

- If we have a S.C condition supported by an existing index we use the index
- If we have a conjunction, such as S.A>5 and S.B <10 with indexes on both, then we can select the better of the two (optimization)
- If there is no index, do we create one?

8

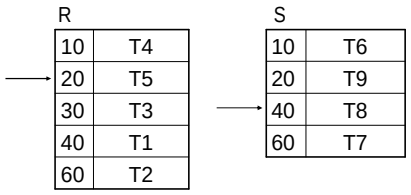
Cost of Joins

- *Nested Loop:* $b_R + |R| \times b_S$ ($b_R$and b_S denote the number of blocks in R and S, resp.) this is the worst case. What is the best case?
- *Pre-existing Index on S:* $b_R + |R| \times k$ (k: depth of index)
- *Block-Nested-Loop Join:* $b_R + b_R \times b_S$ (b_S:number of blocks in S)

9

Sort-Merge Join

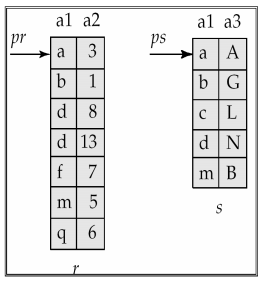
- Sort the relations first and join



10

Merge-Join

1. Can be used only for equi-joins and natural joins
2. Sort both relations on their join attribute (if not already sorted on the join attributes).
3. Merge the sorted relations to join them



11

- Sort-Merge Join (conceptual)

(1) if R and S not sorted, sort them

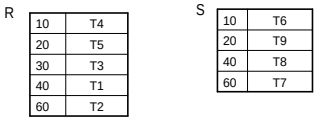
(2) i ? 1; j ? 1;

While (i ? |R|) ? (j ? |S|) do

if R[i].C = S[j].C then outputTuples

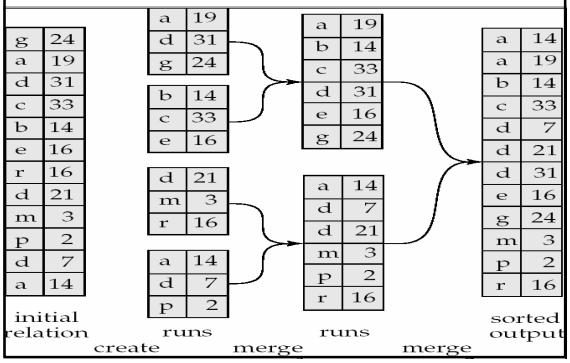
else if R[i].C > S[j].C then j ? j+1

else if R[i].C < S[j].C then i ? i+1



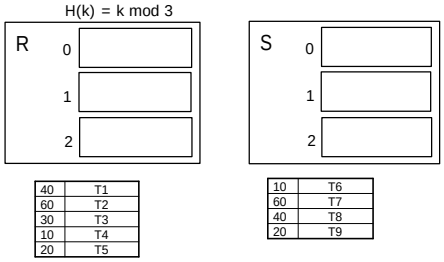
12

External Sorting Using Sort-Merge



Hash Join

- Hash tuples into buckets
- Join between only matching buckets

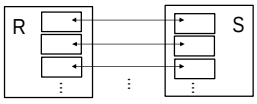


- Hash Join (conceptual)
  - Hash function h, range 1 ? k

Algorithm

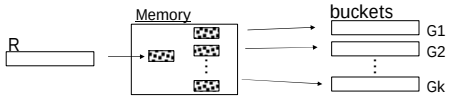
(1) Hashing stage (bucketizing)
- Hash R tuples into G1,…,Gk buckets
- Hash S tuples into H1,…,Hk buckets

(2) Join stage
- For i = 1 to k do
  match tuples in Gi, Hi buckets

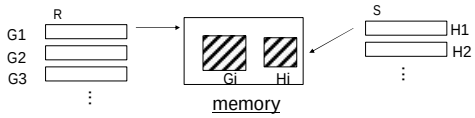


Hash Join

- Step (1): Hashing stage

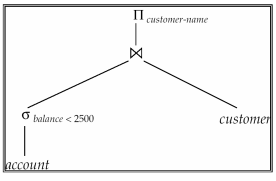


- Step (2): Join stage



Materialization vs Pipelining

- **Materialized evaluation:** evaluate one operation at a time, starting at the lowest-level. Use intermediate results materialized into temporary relations to evaluate next-level operations.
- **Pipelined evaluation:** one tuple at the time.



Summary of Join Algorithms

- Nested-loop join ok for "small" relations (relative to memory size)
- Hash join usually best for *equi-join* if relations not sorted and no index
- Merge join for sorted relations
- Sort merge join good for *non-equi-join*
- Consider index join if index exists
- DBMS maintains *statistics* on data

Statistics collection commands

- DBMS has to collect statistics on tables/indexes
  - For optimal performance
  - Without stats, DBMS does stupid things…
- DB2
  - RUNSTATS ON TABLE <userid>.<table> AND INDEXES ALL
- Oracle
  - ANALYZE TABLE <table> COMPUTE STATISTICS
  - ANALYZE TABLE <table> ESTIMATE STATISTICS (cheaper than COMPUTE)
- Run the command after major update/index construction

19