Class 19: Joins II

Last time

Nested-Loop Joins

Simple
Page-oriented
Block-based

\[(PR \cdot M) \cdot N + M \]  w/ R outer
\[M \cdot N + M \]  w/ k buffer
\[M \cdot N + M \]  w/ k buffer

Index
\[M + M \cdot PR \cdot (\text{index-access-cost} + \text{data-access-cost}) \]

\[\text{hash} \cap \bigtriangledown B^+\text{-tree} \text{ \{clustered\}}\]

Sort-Merge Joins

\[3 \cdot (M + N) \text{ if } B > M \]
\[M + N \text{ if } B > N \]

Today

→ Hash Joins
→ General Join Conditions
→ Aggregates

Hash Joins

→ Use a hash function \( h \) to create partitions of both relations

→ Match tuples only between the corresponding partitions

 Relation \( S \): \( N = 500 \text{ pages} \) \( Pr = 80 \)
 Relation \( R \): \( M = 1000 \text{ pages} \) \( Pr = 100 \)
B buffers  \( R \times S \)

\[ h \text{ hash function} \]

\[ R \ni \text{for } r \in R \]
read \( r \) and add it to buffer \( h(R_i) \)

\[ S \ni \text{for } s \in S \]
read \( s \) and add it to buffer \( h(S_j) \)

for \( l = 1, 2, \ldots, K \)

\[ R \ni \text{for } r \in R \]
read \( r \) and insert into in-memory HT using \( h_2(r) \)

\[ S \ni \text{for } s \in S \]
read \( s \) and probe HT using \( h_2(s) \)
if match found add \( (r, s) \) to the results

clear hash table from memory to proceed
with next pair of partitions

\[ \text{cost} \]
\[ \frac{2 \cdot M}{2 \cdot N} \]
Matching

\[ c_i \xrightarrow{\text{Si}} \]

\[ \text{stream } R_i \xrightarrow{\text{input, output}} \]

read every partition once in memory \( HT \ w/ h2 (\#h) \)

search in \( S_i \) as we stream \( R_i \)

\[ \text{Cost: } M + N \]

Total cost of Hash Join = \( 3 (M + N) = 4500 \Rightarrow 93 \)

Memory Requirements

\[ \rightarrow \text{ enough buffer for the largest partition of the smaller relation } (S) \]

\[ \rightarrow \text{ Input page for the other relation } \]

\[ \rightarrow \text{ Output page } \]

\[ \rightarrow \text{ a few pages of hash metadata } \]

Fudge factor \( f \) (for example \( f = 1.04 \))

If \( h \approx \text{uniform} \)

Size of a partition \( \approx \frac{N}{B-1} \)

\[ B > \frac{fN}{B-1} + 2 \Rightarrow B > \sqrt{fN} \]

What if not enough memory? (for \( S_i \) to fit in memory)

\[ \rightarrow \text{ apply the same algorithm recursively } \]

\[ \rightarrow \text{ read, repartition } S_i, R_i \text{ with } h3 (\#h, \#h) \]

\[ \rightarrow \text{ matching per subpartition (mem. is enough) } \]

\[ \Rightarrow \text{ if not, again recursion } \]
What if we have more memory?

Cost:

- hash S: $N + N - \text{sizeof}(S_1)$
- hash R: $M + M - \text{sizeof}(R_1)$
- marching: $M - \text{sizeof}(R_1) + N + \text{sizeof}(S_1)$

Total: $3(M+N) - 2(\text{sizeof}(S_1) + \text{sizeof}(R_1))$

$B = 300$

$M = 1000$

$N = 500$

If $B < 600$

read S once + build hash table

scan R once ; prob S on-the-fly
Hash Join vs SMJ

\[
\begin{align*}
\text{cost} & \quad 3(M+N) \\
\text{mem. req} & \quad B^D \frac{\ell}{\sqrt{N}} \leq \text{smaller} \\
& \quad B^D \sqrt{104.586} = 23 \\
B(2m.r) \leq N & \quad 3(M+N) - 2(\text{sizeof}(R_2) - \text{sizeof}(S_2)) \\
B > N & \quad M+N \\
\text{output} & \quad \text{if input sorted} \quad 3(M+N) \\
\text{BUT} & \quad \text{sensitive to data skew}
\end{align*}
\]

- (a) equality joins on several attributes
- (b) inequality joins

- (a) for INLJ we need index with all attributes in join conditions
  - sort/hash use combination of all attributes
- (b) INLJ w/ B^+.Tree (not Hash Index)
  - HS/SMJ cannot work
  - Bloor NLS the best approach

Set

\[
\begin{align*}
\text{UNION/EXCEPT (set difference)} \\
\text{sort S+R on all attributes} \\
\text{merging/sorting} & \quad \text{discard duplicates (UNION)} \\
\text{set difference}
\end{align*}
\]
→ hashing
→ partition R+S
→ S-part probe corr. R-part
→ discard duplicates (UNION)
→ set-difference
→ Intersection → special case of Join

Intersection special case of Join
Equality across all attributes

Aggregation

→ SELECT AVG(sal) FROM E
→ SCAN once

→ GROUP BY
  L age, avg_salary)

hash (age) → <age, salary, count>

sort (age) calculate "running info" of aggregation on-the-fly

→ if we have an index on <Group-by, select, where>
can use only the index [WAY FASTER]

→ Buffering
  Among things in parallel
tough to estimate what is asked by BP

SNLS B2N √
BN LRU→sequential flooding.
MRU √

INLS § sort the outer relation