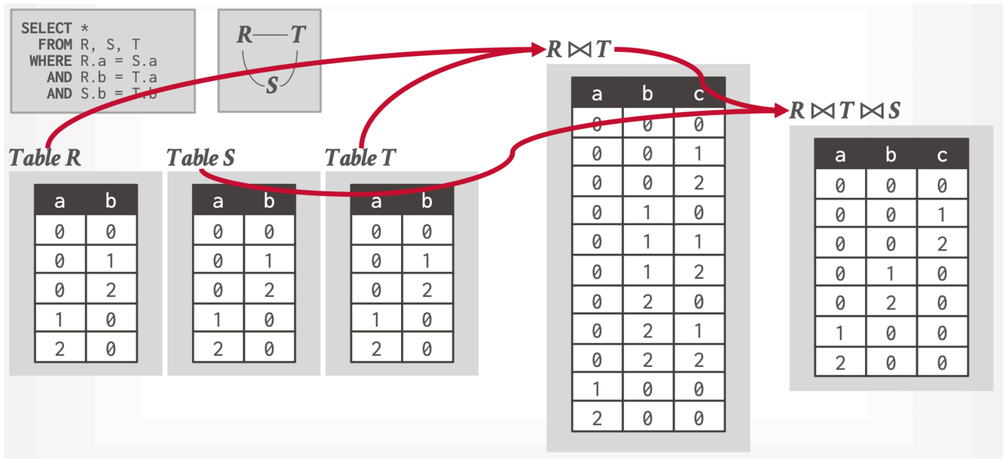
Lecture #10: Multi-Way Join Algorithms

15-721 Advanced Database Systems (Spring 2024)
`https://15721.courses.cs.cmu.edu/spring2024/`
Carnegie Mellon University
Prof. Andy Pavlo

1 Introduction

Binary joins are ubiquitous in most relational queries and their optimizations are well-studied. However, they are not efficient if the intermediate result is significantly larger than the final result. This is especially problematic for join queries for graph databases.



```
SELECT *
  FROM R, S, T
 WHERE R.a = S.a
   AND R.b = T.a
   AND S.b = T.b
```

Table R

a	b
0	0
0	1
0	2
1	0
2	0

Table S

a	b
0	0
0	1
0	2
1	0
2	0

Table T

a	b
0	0
0	1
0	2
1	0
2	0

R ⋈ T

a	b	c
0	0	0
0	0	1
0	0	2
0	1	0
0	1	1
0	1	2
0	2	0
0	2	1
0	2	2
1	0	0
2	0	0

R ⋈ T ⋈ S

a	b	c
0	0	0
0	0	1
0	0	2
0	1	0
0	2	0
1	0	0
2	0	0

For example, in the query described above, joining any combination of two tables produces a larger intermediate result than the inputs and the final output. To overcome this limitation, we can use multi-way join algorithms that execute multiple joins as a whole.

1.1 Worst-Case Optimal Joins (WCOJ)

Worst-case optimal joins provide a theoretical criterion of how multi-way joins perform well on arbitrary inputs. The exact definition of WCOJ is as follows.

Definition The runtime of the join algorithm is better than all other join algorithms when the query and data represent the worst possible scenario.

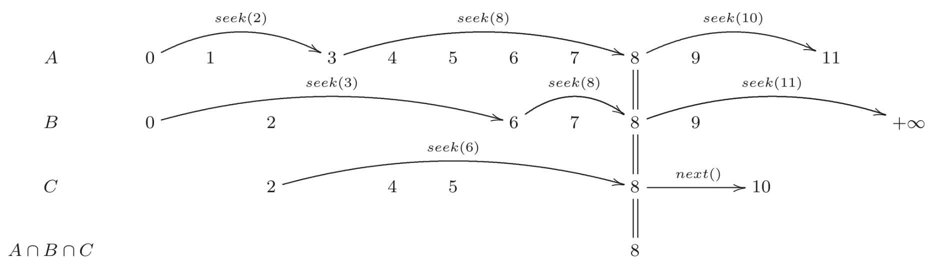
WCOJ algorithms have runtime bounded by the size of the final output, not by the size of each input table. This can be done by performing join by examining a variable at a time instead of examining a relation at a time.

2 Leap-Frog Trie Join

Leap-Frog Trie Join satisfies the complexity bound requirement of WCOJ by finding the matching key values using trie data structure [3].

2.1 Common Matching

First of all, the algorithm sorts each table by the join key and tries to find pairs of tuples that have a common join key.



For tables A, B, and C which consist of described join key values, we create three iterators for each table that starts with (0, 0, 2). Then,

- Because C is positioned at 2, A should skip next until it finds the first element which is larger than or equal to 2, and it stops at 3.
- Given A is now at 3, we should then position B to the next key larger than or equal to 3, and it goes directly to 6.
- Given B is at 6, C seeks the next position and reaches 8.
- Then we position A to the next position, and it also reaches 8.
- B also reaches 8 by advancing the iterator.

Consequently, we found the first common matching: 8. The algorithm moves on to find the next common matching tuple pair.

2.2 Trie Index

Leap-Frog Trie Join uses a trie index data structure to accelerate the common matching.

For each relation, we locate the attributes used at least once in the entire multi-join. Then, these attributes are used as a key to construct a trie. Each level of the index represents a single attribute in the key. The order of the attributes in each table should be globally consistent among the tables.

2.3 Join Algorithm

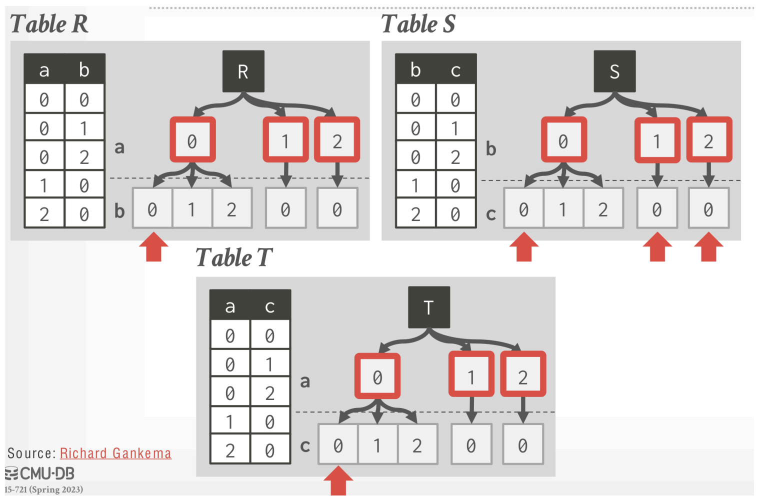


Table R

a	b
0	0
0	1
0	2
1	0
2	0

Table S

b	c
0	0
0	1
0	2
1	0
2	0

Table T

a	c
0	0
0	1
0	2
1	0
2	0

To join tables R, S, and T, we do the following steps:

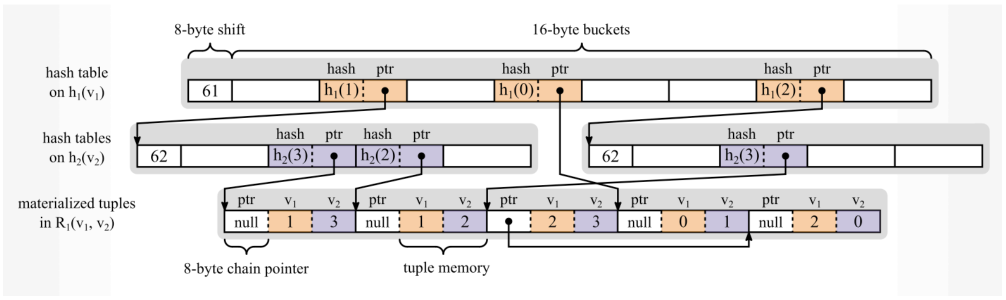
- First, we select the key a for iteration. For tables R and T which have key a, we use the matching algorithm described above.
- Once we find a matching value a' for key a, we then start examining the second key b in tables R and S, using the bottom level of R's trie and the top level of S's trie.
- After we find a matching value b' for key b, we then examine the third key c in tables S and T to obtain matching value c'.
- Generate an output tuple (a', b', c') and repeat the process.

2.4 Limitation

Building a trie index for all relations on the fly is expensive. Even if we build them in advance for read-only tables, the global order of attribute matching should be predetermined because this global order decides the order of attributes in each trie. Hence, if we want to select the global order during runtime, for each table, we should build trie indexes for all possible permutations of join attributes, which is impractical.

3 Hash Trie Join

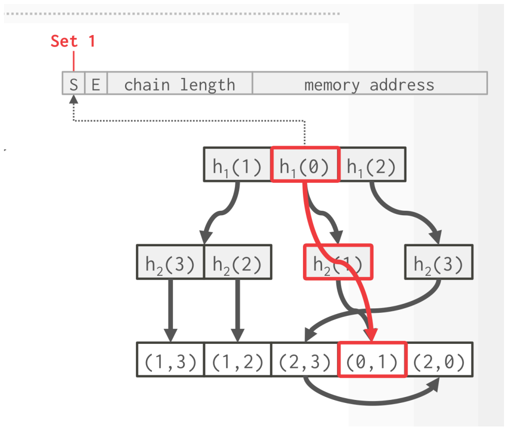
Hash Trie Join overcomes this limitation by storing the hash value of keys instead of storing the entire key in the trie [1]. Each bucket points to the lower level of the trie or the tuple.



For optimization, the 64-bit pointer includes metadata in the upper 16 bits because x64 machines only use the lower 48 bits and ignore the upper part.

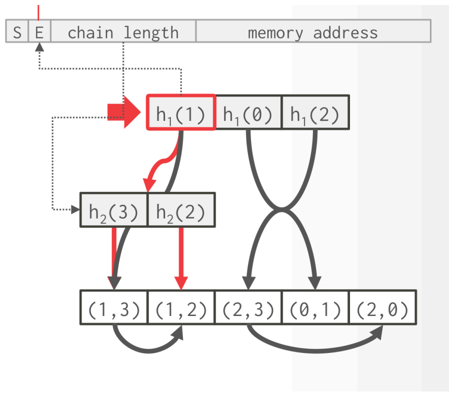
3.1 Singleton Pruning

The lower level of the hash trie usually has fewer data. Therefore, we can directly jump to the final layer to avoid allocating singleton nodes, as `h(0)` shown in the below figure.



3.2 Lazy Child Expansion

Some trie nodes may never be accessed if the filter selectivity is high. Therefore, we can defer materializing inner nodes until they are accessed during the probe phase.



3.3 Query Optimizer

Employing multi-way join algorithms to the end-to-end DBMS requires further considerations.

Multi-way joins are slower than binary joins if the query's intermediate results are not larger than its inputs. Therefore, for a given query plan, the DBMS should decide whether or not it will use the multi-way join algorithm. This is achieved by extending the query optimizer to use heuristics to select the appropriate join algorithm.

4 Other Optimizations

4.1 Factorization

Instead of materializing duplicate tuples in the intermediate result, we can choose to keep the counter for the number of duplicates to reduce the output size. This requires rewriting database codes to be aware of the meaning of counter values [2].

References

[1] M. Freitag, M. Bandle, T. Schmidt, A. Kemper, and T. Neumann. Adopting worst-case optimal joins in relational database systems. *Proceedings of the VLDB Endowment*, 13(12):1891–1904, 2020.

[2] D. ten Wolde, T. Singh, G. Szárnyas, and P. Boncz. Duckpgq: Efficient property graph queries in an analytical rdbms. In *Proceedings of the Conference on Innovative Data Systems Research. https://www. cidrdb. org/cidr2023/papers/p66-wolde. pdf*, 2023.

[3] T. L. Veldhuizen. Leapfrog triejoin: a worst-case optimal join algorithm. *arXiv preprint arXiv:1210.0481*, 2012.