Lecture #13

Carnegie Mellon University

ADVANCED DATABASE SYSTEMS

Multi-Way Join Algorithms

Andy Pavlo // 15-721 // Spring 2023

MOTIVATION

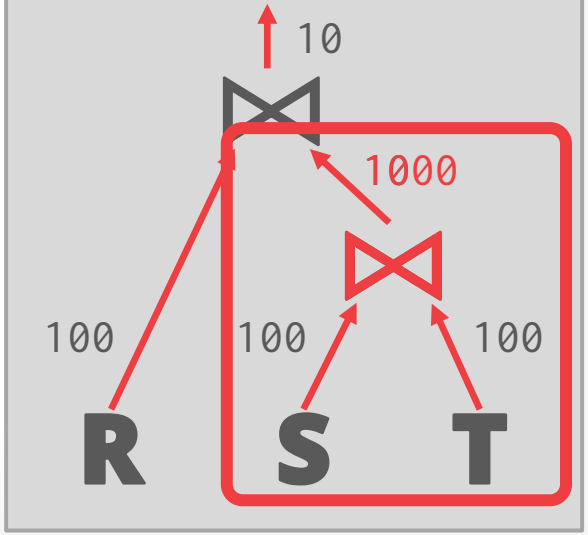
Binary (i.e., two table) joins are ubiquitous in relational DBMSs.

→ Decades of research to make these algorithms highly efficient.

This is the optimal approach when the output of the join is smaller than its two inputs.

But things can go bad…

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



MOTIVATION

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

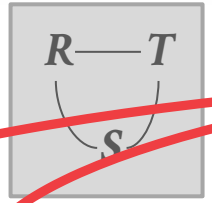


Table R

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

$R \bowtie T$

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

$R \bowtie T \bowtie S$

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

Wasted Computation
Wasted Storage

TODAY'S AGENDA

Worst-Case Optimal Joins

Leap-Frog Trie Join

Hash Trie Join

WORST-CASE OPTIMAL JOINS

Perform join by examining a variable at a time instead of a relation at a time.

→ First considered by other Germans in 2008.

The runtime of a WCOJ algorithm is bounded by the output size of the result and depends on the variable evaluation ordering.

→ If more tuples match in the intermediate results, then the DBMS must check the other tables.

The more tables a WCOJ algorithm joins, the better its performance relative to the input.

WORST-CASE OPTIMAL JOINS

Alternative Definition #1: The worst-case runtime of the algorithm meets a known lower bound for the worst-case runtime of any join algorithm.

Alternative Definition #2: The runtime of the join algorithm is better than all other join algorithms when the query and data represent the worst possible scenario.

Source: Semih Salihoğlu

WORST-CASE OPTIMAL JOINS

As of 2023, very few DBMSs support worst-case optimal join algorithms.

→ First known implementations were in LogicBlox and EmptyHeaded (Stanford)

These joins will be more common because the SQL 2023 standard includes property graph query extensions (SQL/PGQ).

RelationalAI

UMBRA

DuckDB

LogicBlox

KÙZU

SKEW STRIKES BACK: NEW DEVELOPMENTS IN THE THEORY OF JOIN ALGORITHMS
SIGMOD RECORD 2013

LEAP-FROG TRIE JOIN

Relations must be indexes (or sorted) on the join keys in advance before performing the join.

Represent relations with multiple attributes as tries.

→ One trie per relation.

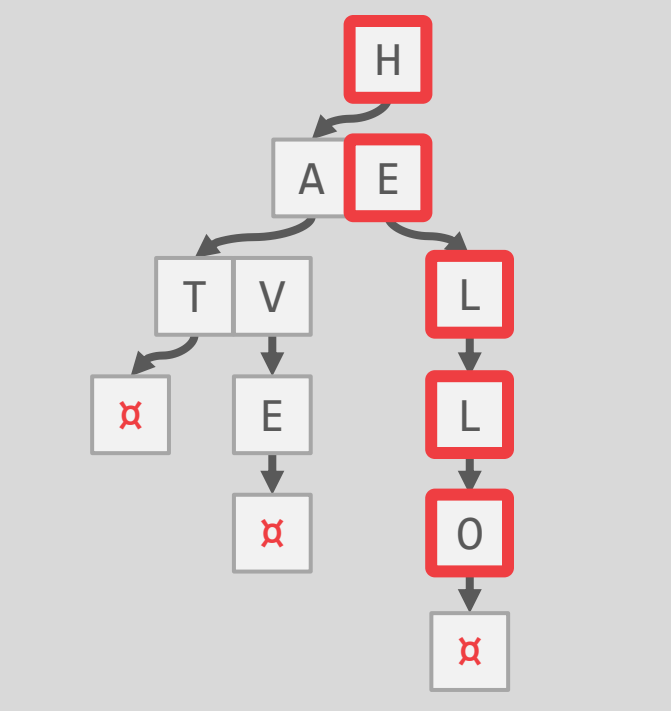
→ Each level in the trie represents a single attribute in the join keys.

Developed by LogicBlox in the early 2010s.

LEAPFROG TRIEJOIN: A SIMPLE, WORST-CASE OPTIMAL JOIN ALGORITHM
ICDT 2014

TRIE

Keys: HELLO HAT, HAVE



Use a digital representation of keys to examine prefixes one-by-one instead of comparing entire key.

→ Also known as ***Digital Search Tree***, ***Prefix Tree***.

Contrast this with a B+Tree that stores all the digits of a key together in a node.

LEAP-FROG TRIE JOIN

Table X

id
0
1
3
4
5
6
7
8
9
10

SORT!

Table Y

id
0
2
6
7
8
9

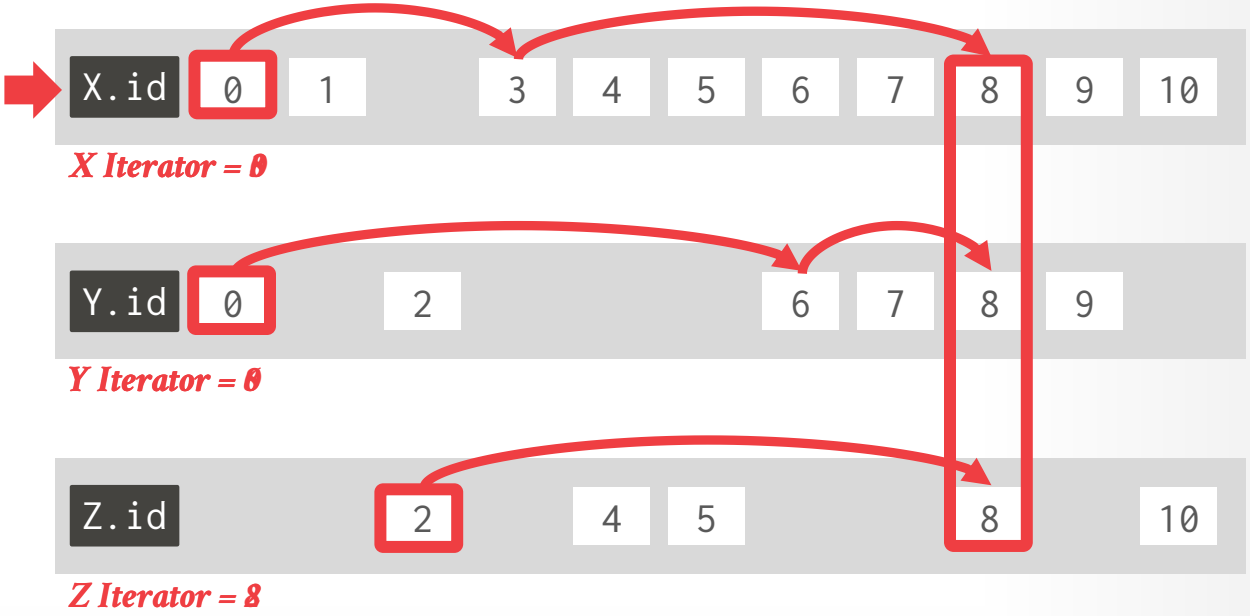
SORT!

Table Z

id
2
4
5
8
10

SORT!

$X \bowtie Y \bowtie Z$

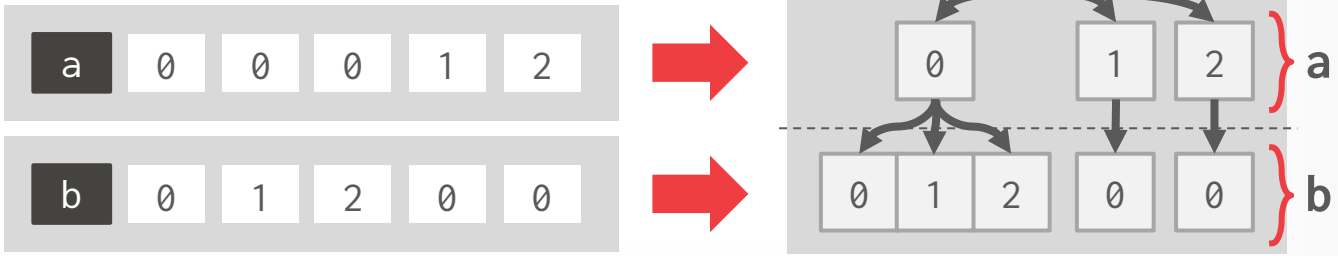


LEAP-FROG TRIE JOIN

Represent relations with multiple attributes as tries where each attribute is a different level in the data structure.

Table R

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



Trie

LEAP-FROG TRIE JOIN

Table R

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

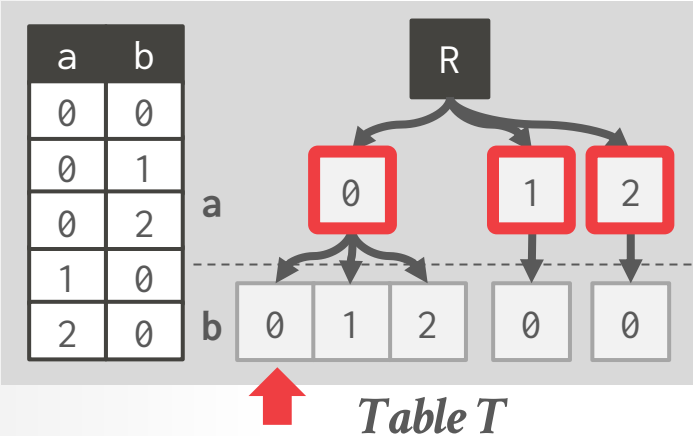


Table S

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

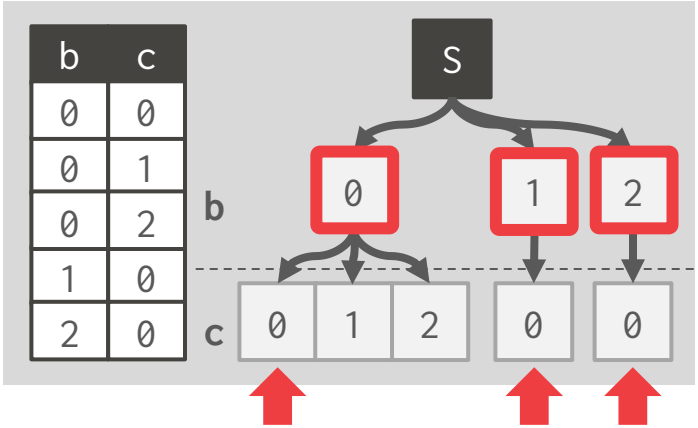
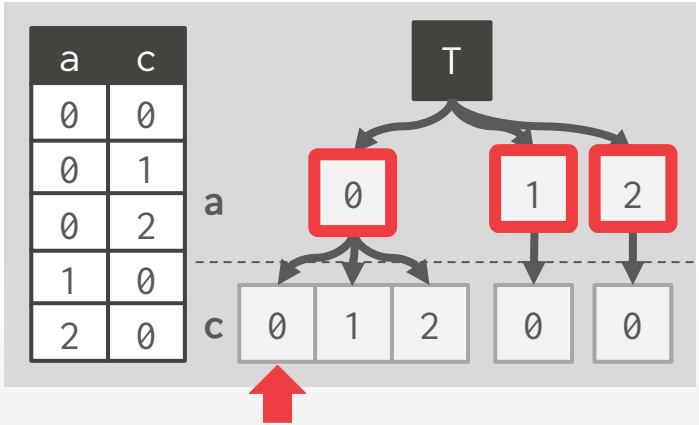


Table T

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



$S.c \rightarrow (0,1,2)$

$\cap$

$T.c \rightarrow (0,1,2)$

$R \bowtie S \bowtie T$

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

Source: Richard Gankema

OBSERVATION

Building a trie for every relation on the fly is expensive.

→ Even if the database is read-only, building tries for every possible join ordering is impractical.

An alternative approach is to use nested hash tables, but this is also expensive:

→ At least one key comparison to detect hash collisions.

→ Need to store the actual keys or pointers to the tuples to deal with collisions. Lots of cache misses.

→ Need to deal with variable-length keys using dictionary encoding.

MULTI-WAY HASH TRIE JOINS

Instead of storing the join keys in nested hash tables, store hash values.

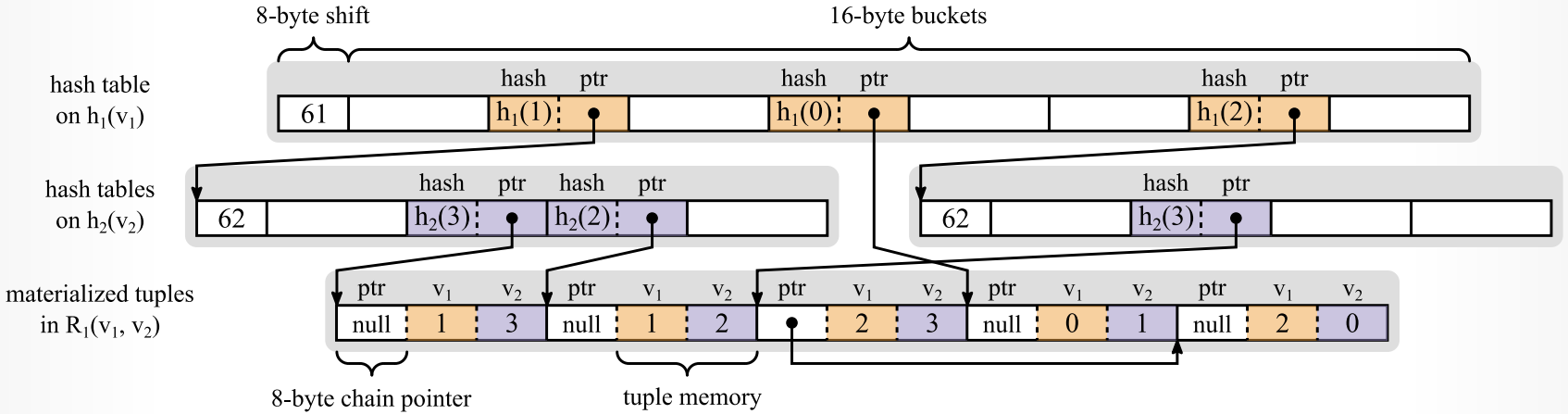
No type-specific logic for computing set intersections and lookup operations.

→ DBMS only uses fast integer comparisons.

→ Need to remove false positive matches on hash collisions.

ADOPTING WORST-CASE OPTIMAL JOINS IN RELATIONAL DATABASE SYSTEMS
SIGMOD 2020

HASH TRIE



Source: Michael Freitag

HASH TRIE: TAGGED POINTERS

Exploit unused portion of 64-bit pointers to trie nodes to store additional meta-data.

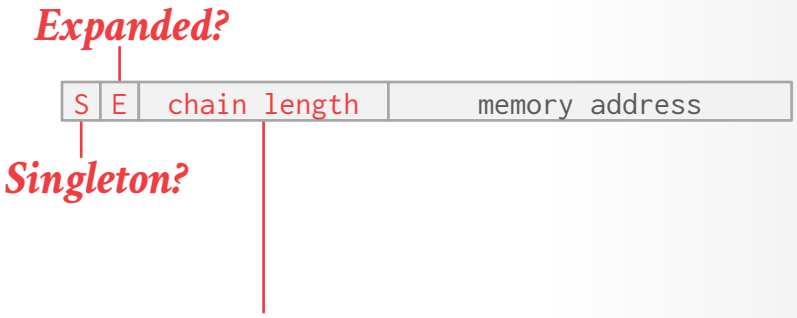
→ x86-64 only uses 48 bits for memory addresses.

1-bit: Singleton Flag

1-bit: Expansion Flag

14-bits: Chain Length

48-bits: Memory Address

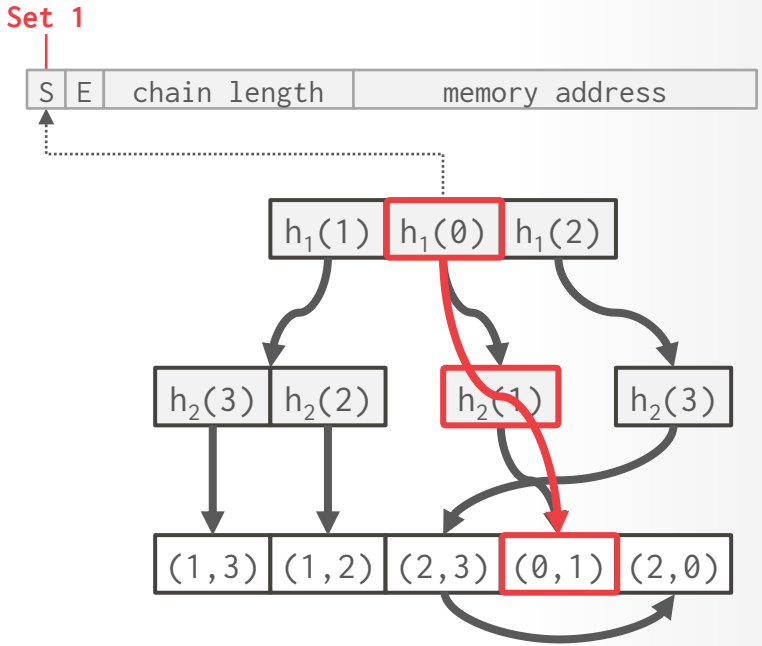


HASH TRIE: SINGLETON PRUNING

The size of hash tables in the trie get smaller in the lower trie levels.

→ Entries in inner nodes often point to only a single tuple below it.

Optimization: Instead of storing each level of trie for these paths, only store a pointer directly to the singleton tuple.

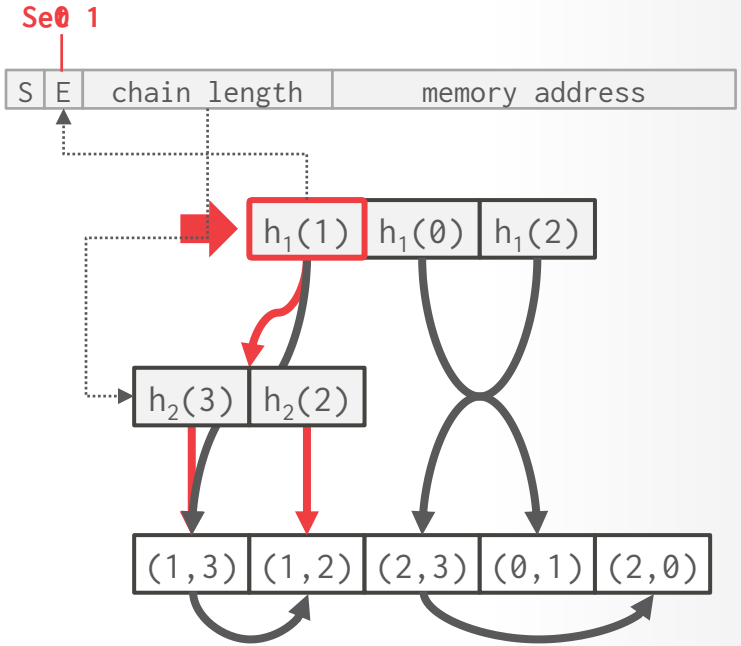


HASH TRIE: LAZY CHILD EXPANSION

Depending on the selectivity of the intersection operation, the DBMS never accesses many inner nodes.

Optimization: Only materialize inner nodes in the trie when they are accessed during the probe phase.

→ Must always create root node.



OBSERVATION

Multi-way joins are slower than binary joins if the query's intermediate results are not larger than its inputs.

Umbra extended their optimizer to use heuristics to decide whether to use binary join vs. WCOJ.

PARTING THOUGHTS

Multi-way joins are an active area of research.

In the next decade they could be the default choice for supporting joins in a relational DBMS.

This will be the deathblow to graph DBMSs.

NEXT CLASS

Converting UDFs into inline SQL/RA

→ It's an amazing idea.

→ But Sam has ruined the dream for me.

→ This is why we can't have nice things…