RDBMS Hash Joins

- Only works for equijoins

1. Fundamental concept: *Hash partitioning*

1. First: Do alg. for conventional RDBMS
2. Then: Massively parallel version

//(just the *golden case*)

Given relations L, R

Let Hash(JK) be a virtual column, presented for illustrative purposes.

Hash(JK) → { 0 ,1}

L ⋈ R

L

PK	JK	a	Hash (JK)
1	Dan	A	0
2	Joe	B	0
3	Bob	C	1

JK

R

PK	JK	b	Hash (JK)
4	Dan	D	0
5	Bob	E	1
6	Dan	F	0

=

PK L	PK R	JK	a	b
1	4	Dan	A	D
1	6	Dan	A	F
3	5	Bob	C	E

Hash Join 1

Hash Partition on Hash(JK)

hash partition

L

R

0

PK	JK	a	Hash (JK)
1	Dan	A	0
2	Joe	B	0

PK	JK	b	Hash (JK)
4	Dan	D	0
6	Dan	F	0

1

PK	JK	a	Hash (JK)
3	Bob	C	1

PK	JK	b	Hash (JK)
5	Bob	E	1

Hash Join 2

Join Data Within A Partition

partition

L

R

0

PK	JK	a	Hash (JK)
1	Dan	A	0
2	Joe	B	0

PK	JK	b	Hash (JK)
4	Dan	D	0
6	Dan	F	0

=

PK L	PK R	JK	a	b
1	4	Dan	A	D
1	6	Dan	A	F

1

PK	JK	a	Hash (JK)
3	Bob	C	1

PK	JK	b	Hash (JK)
5	Bob	E	1

=

PK L	PK R	JK	a	b
3	5	Bob	C	E

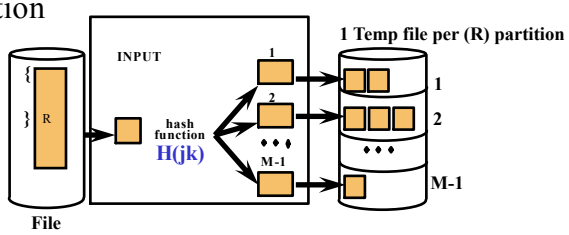
Union of the result of the two partitions yields correct

Hash Join
Conventional RDBMS

- Similar to two phase multiway merge join
- Phases are reversed

Phase 1, Hash Partition R

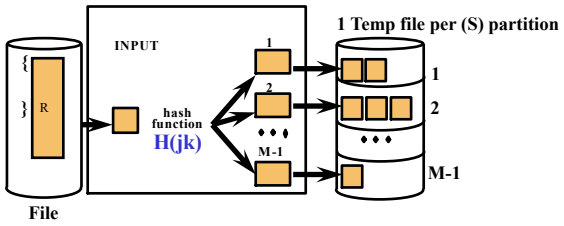
- Read 1 block of R into memory, at a time
- One buffer, Max of M-1 buffers, for each partition



- As buffers file, flush to temp files

Phase 1, Hash Partition s

- Simply repeat

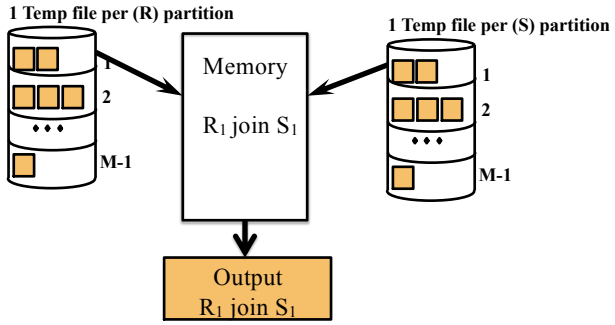


Hash Join

- Phase 1, hash partition, (1 block at time)
  - read R, and write out hash buckets
  - read S, and write out hash buckets
- Phase 2, join buckets, (1 pair hash buckets at a time)
  - For each hash value
    - read corresponding pair of R, S, hash buckets
    - join the buckets, write results.

Phase 2, Join Pairs of Partitions

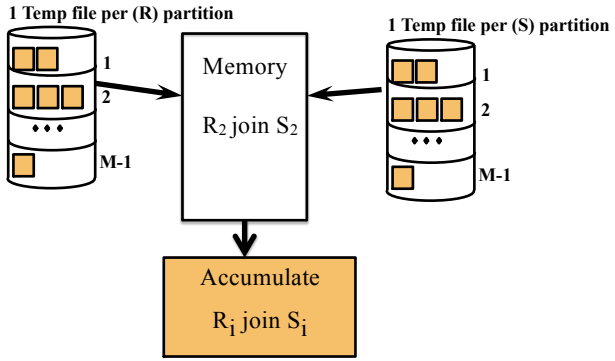
partition 1



Phase 2, Join Pairs of Partitions

partition 2

… till done



How many partitions?

- Really want the contents of a partition to fit into memory?

  (Why?)

- What if it doesn't
  - Resort to block nest loops

  Or

  - Recursively hash the offending partition
    - - Use a different hash function? (Why?)

Cost of Hash Joins

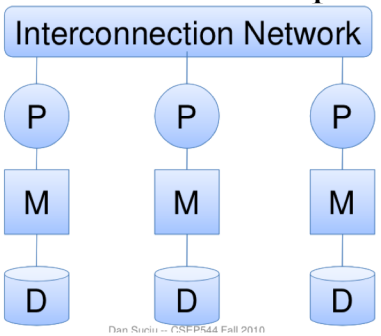
For each relation

- read it,
- hash
- write

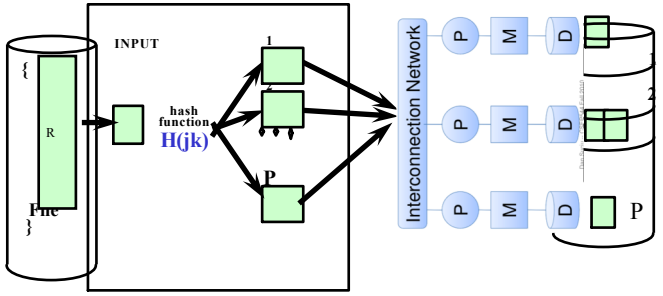
Read pair of hash buckets

Cost = 3 (B(R) + B(S))

How do we do this in parallel?

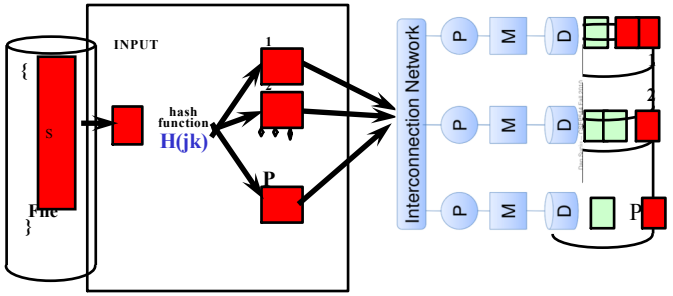


Phase 1, Hash Partition R

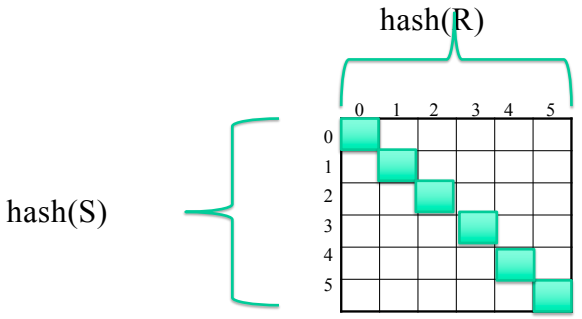


What is P?

Phase 1, Hash Partition S



Classic Figure Illustrating *Grace Join*



But what if it's not an equijoin?

- Nested loop joins have their place in the world
  - both by function
  - by detailed cost.

Hybrid-Hash Join R join S

S

- Hash smaller relation, S, first
- for k hash buckets - *do not* divide memory evenly
  - for M buffers assign // ignore output
    - M-k buffers to one bucket, β
    - 1 buffer for each of remaining k-1 buckets
    - 1 buffer remains "for R"
- Hash S, writing out the "size 1 buffers" as they fill

R

- Hash R into k buckets
- If tuple hashes to β, do the join,
  - otherwise write to one of the k-1 buckets per hash
  - write out as they fill
- Complete a hash join

Many Join Algorithms

A Comprehensive Survey of Join Techniques in Relational Databases
Yuping Yang, Mukesh Singhal
ACM Surveys **(1997)**

.