

Simple Hash Join

R

S



$\text{JP}(\mathbf{r}, \mathbf{s}) := \mathbf{r.x} == \mathbf{s.x}$

//definition of the join predicate

SimpleHashJoin(R, S, JP(r,s)):

 indexOnRX := build_ht(R.x);

//build hash table on R.x

↑
key_i → v

Simple Hash Join

R

S

JP(r,s) := r.x == s.x

//definition of the join predicate

SimpleHashJoin(R, S, JP(r,s)):

 indexOnRX := build_ht(R.x);

//build hash table on R.x

 ForEach s in S:

//for every tuple in S

 queryResultSet = indexOnRX.query(s.x);

//query index for this s.x (aka probe the index)

↑
point-query

Simple Hash Join

R

S

JP(r,s) := **r.x** == **s.x**

indexOnRX := catalog.get(indexes, **R.x**);

//definition of the join predicate

//use existing index on R.x

= hash table

Simple Hash Join

R

S

JP(r,s) := r.x == s.x

indexOnRX := catalog.get(indexes, R.x);

SimpleHashJoin(indexOnRX, S, JP(r,s)):

~~// indexOnRX := build_ht(R.x);~~

//definition of the join predicate

//use existing index on R.x

//precondition for this join algorithm

~~//build hash table on R~~

Simple Hash Join

R

S

JP(r,s) := r.x == s.x

indexOnRX := catalog.get(indexes, R.x);

SimpleHashJoin(indexOnRX, S, JP(r,s)):

indexOnRX := build_ht(R.x);

ForEach s in S:

queryResultSet = **indexOnRX**.query(s.x);

If queryResultSet NOT empty:

 output({s} × **queryResultSet**);

//definition of the join predicate

//use existing index on R.x

//precondition for this join algorithm

//build hash table on R

//for every tuple in S

//query index for this s (aka probe the index)

//did the query return results?

//output join results

No Difference : SHJ and INLJ

except : point in time when we create index

index type : hash table vs. any index

SHJ = special case
of INLJ

Index Nested-Loop Join

R

S

JP(r,s) := $r.x == s.x$

indexOnRX := catalog.get(indexes, **R.x**);

IndexNestedLoopJoin(indexOnRX, S, JP(r,s)):

ForEach s in S:

queryResultSet = **indexOnRX**.query(**s.x**);

If **queryResultSet** NOT empty:

 output({**s**} × **queryResultSet**);

//definition of the join predicate

//use existing index on R.x

//precondition for this join algorithm

//for every tuple in S

//query existing index on R.x for this s.x

//did the query return results?

//output join results