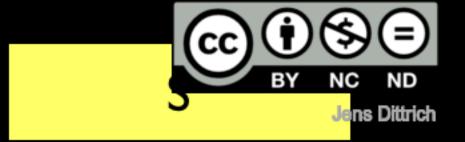
R



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

//definition of the join predicate

//build hash table on R.x

R

```
//definition of the join predicate
```

```
//build hash table on R.x
//for every tuple in S
//query index for this s.x (aka probe the index)
```

```
JP(r,s) := r.x == s.x
indexOnRX := catalog.get( indexes, R.x);
```

R

//definition of the join predicate //use existing index on R.x =
$$\lambda_{er} L + \lambda_{el} l_e$$

```
R
```

R

S

```
JP(r,s) := r.x == s.x
                                                             //definition of the join predicate
indexOnRX := catalog.get( indexes, R.x );
                                                             //use existing index on R.x
                                                             //precondition for this join algorithm
SimpleHashJoin(indexOnRX, S, JP(r,s)):
                                                             //build hash table on R
    indexOnRX := build_ht(R.x);
    ForEach s in S:
                                                             //for every tuple in S
        queryResultSet = indexOnRX.query(s.x);
                                                             //query index for this s (aka probe the index)
        If queryResultSet NOT empty:
                                                             //did the query return results?
            output( {s} × queryResultSet );
                                                             //output join results
                                                                                = ZN6c:5(
      No Difference: SHJ and INLJ
               except: point in time when
        inter type: Lost tolle us. or; inter
```

Index Nested-Loop Join

```
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 );
```

```
R
```

```
//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
```

//definition of the join predicate