Implementing Relational Algebra Operators in a DBMS

Project #3

Relational Algebra

- Operations that can be performed on sets to perform queries in a relational model
- Basic set operations (Union, Intersection, Difference, etc)
- Unary operators
  - *Selection* – filter a set based on some predicate
  - *Projection* – Select only specific attributes of a set
    - In a DBMS, by default any duplicate tuples in the result of a Projection operation are not removed
- Binary operators
  - *Cartesian Product* – all possible pairings between tuples in two sets
  - *Join* – Conceptually, a Cartesian Product followed by a Selection operation, but there are more efficient algorithms available.

Relational Algebra and a DBMS

- When a DBMS parses an incoming SQL query, it eventually transforms it into a Relational Algebra query for evaluation – *a Query Tree/Query Evaluation Plan (QEP)*
- Each Relational Algebra operator is implemented using a common Iterator interface that will return one tuple from the result set at a time
- Allows for *lazy evaluation*

  Each Iterator is only accessed when it is required to return the next tuple
- Allows for *pipelining*

  Tuples can "flow" through the query one-at-a-time without waiting for any operation to complete, giving immediate results to the query

Iterator Interface

- **`open()`**
  - Creates and opens the Iterator
  - In terms of *Project 3*, this is the Constructor for the Iterator object
- **`close()`**
  - Destroys the Iterator, releasing any resources
  - In terms of *Project 3*, this will recursively close() any input Iterators
- **`restart()`**
  - Restarts the Iterator, as if it had just been opened
  - If the Iterator is a subtree in the Query Tree instead of a base scan, this will recursively restart the entire subtree
- **`hasNext()`**
  - Returns true if there is another tuple in the result, false otherwise
  - Not always sufficient to simply recursively call **`hasNext()`** on the input Iterators to compute this (e.g. *Selection, Joins*)
- **`getNext()`**
  - Returns the next tuple in the result

Relational Algebra Iterators: Minibase

- Basic Scans
  - Iterate through the base relational table (*FileScan*) or through an index defined on the table (*IndexScan* for the entire index, or *KeyScan* for only the tuples matching a given key)
- Operators
  - *Projection*.
  - *Selection*.
  - *SimpleJoin* (nested loop join – already implemented).
  - *HashJoin* (use hashing to perform a *Hash Join*).
- Queries are executed by starting with the root Iterator in the Query Tree and recursively calling `hasNext()` and `getNext()`.

Nested Loop Join Example

- **SQL**

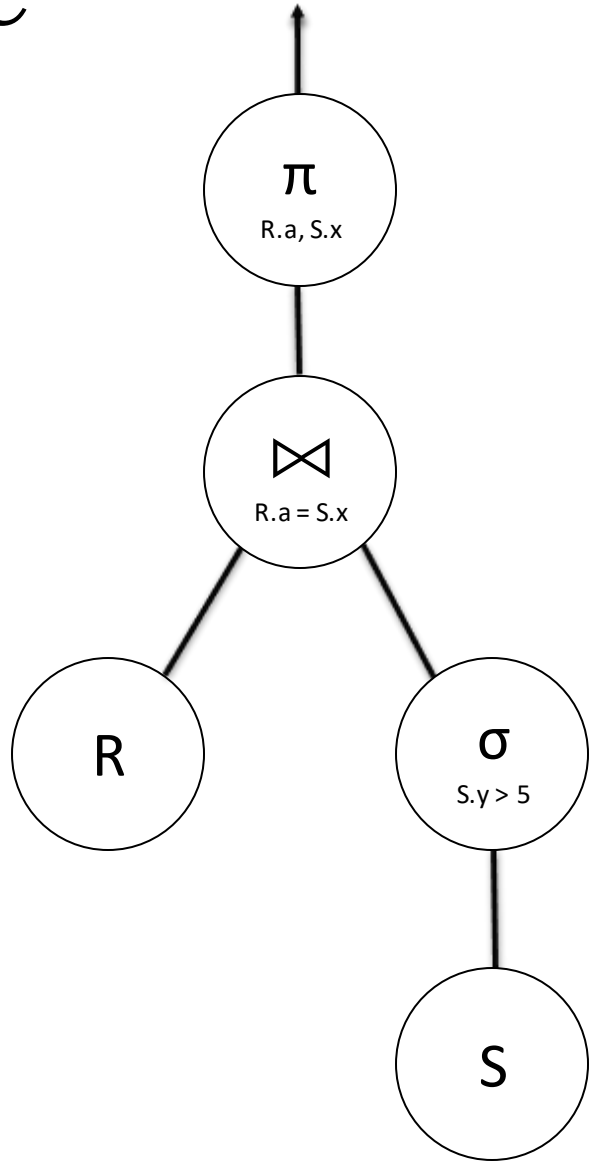
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

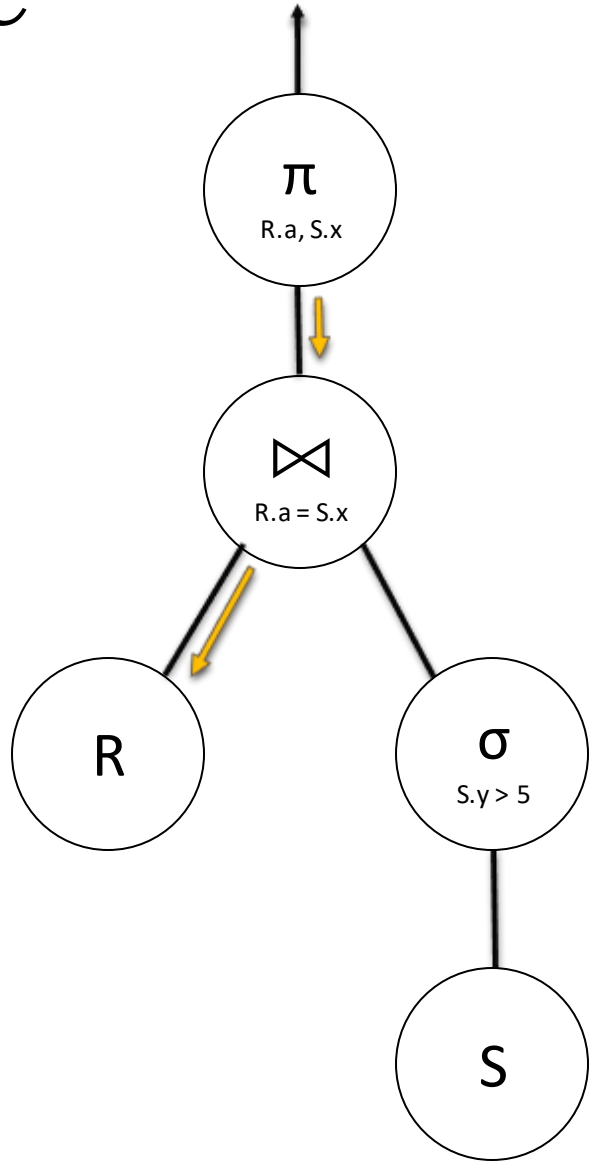
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

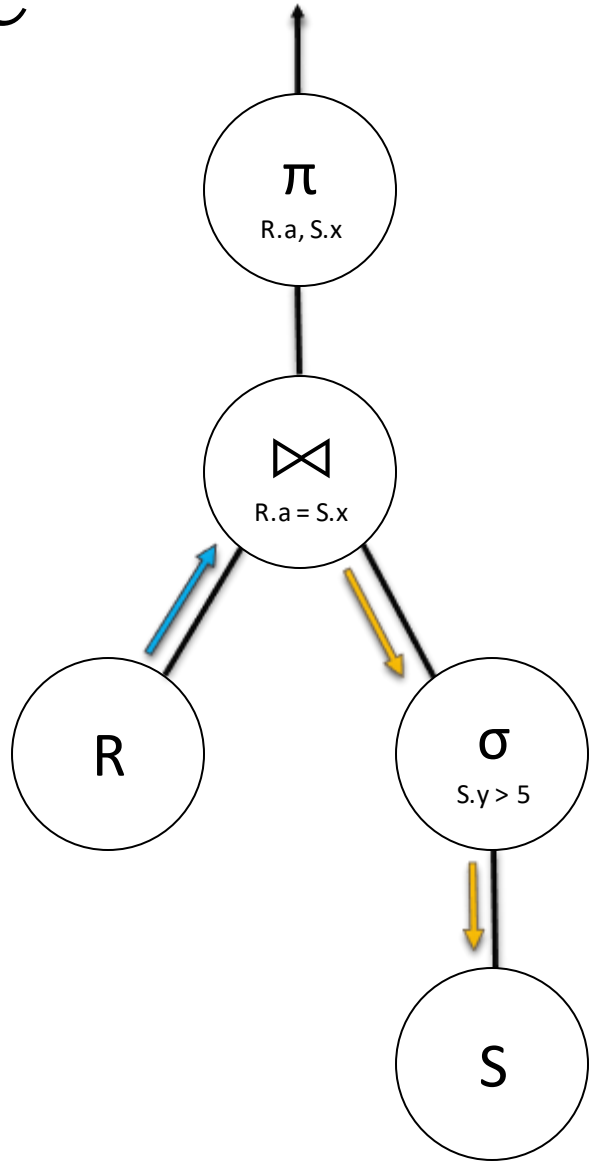
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

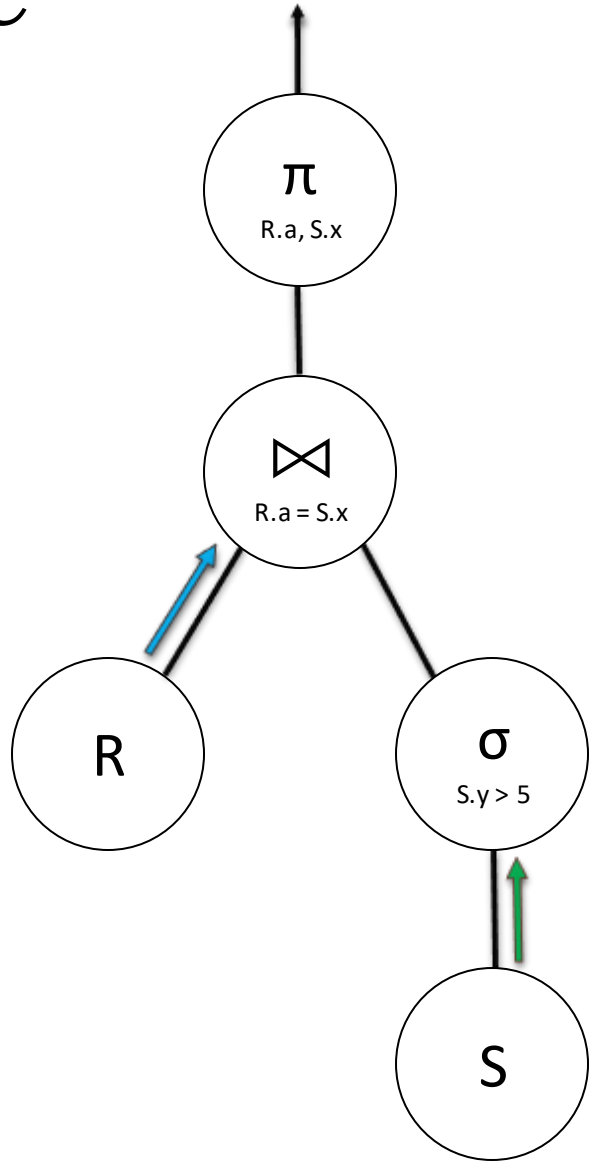
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

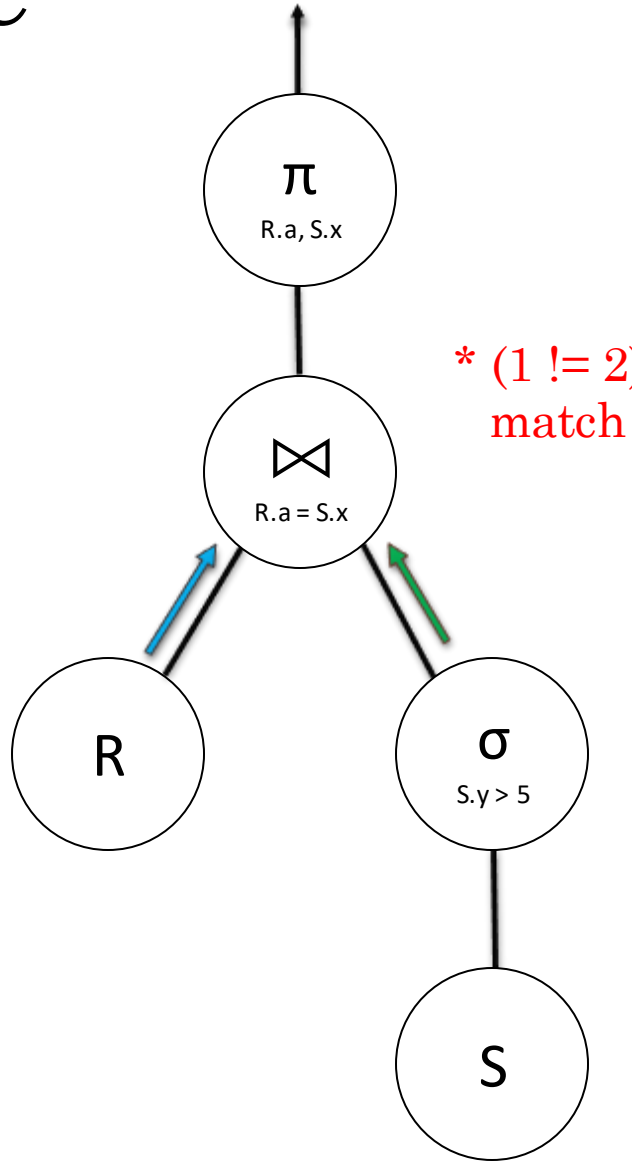
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



* (1 != 2) Keys do not match

Nested Loop Join Example

- **SQL**

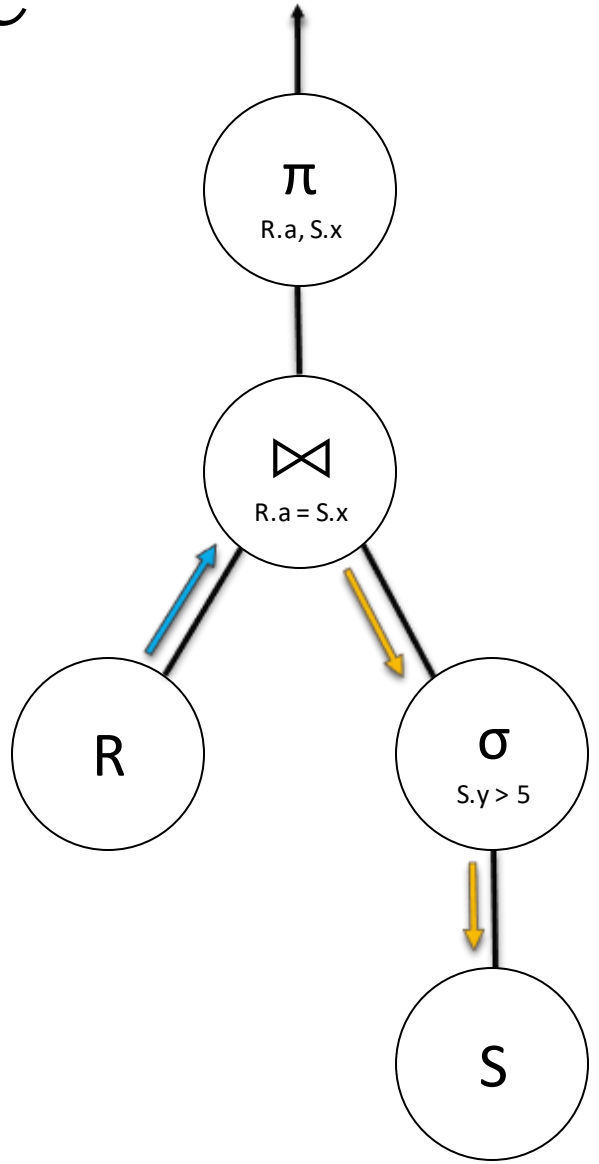
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

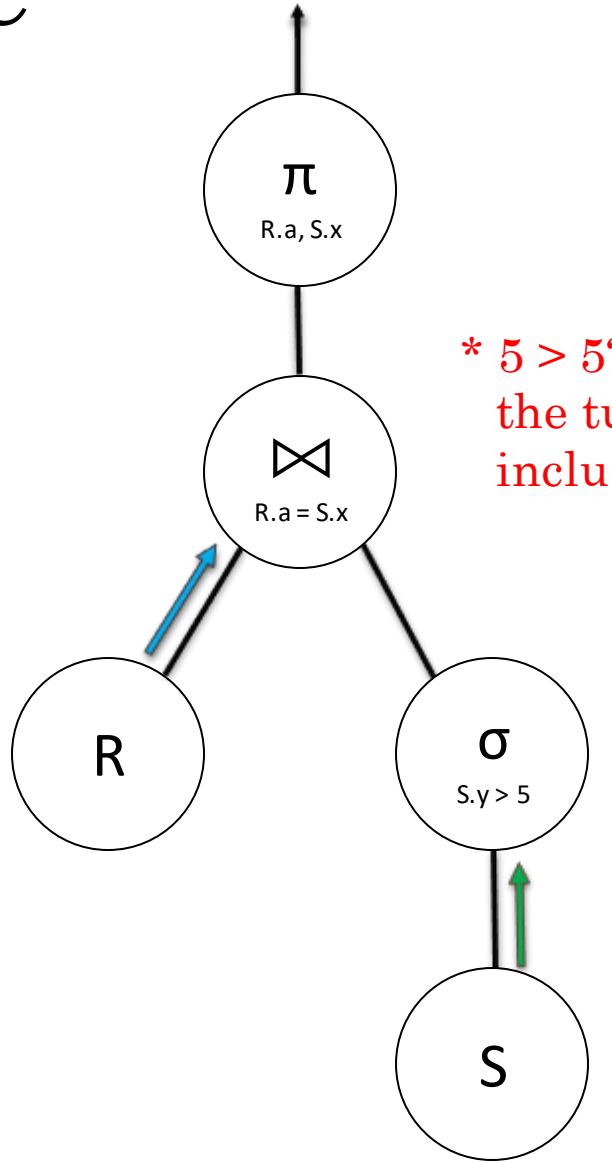
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



* 5 > 5? No, therefore the tuple is not included

Nested Loop Join Example

- **SQL**

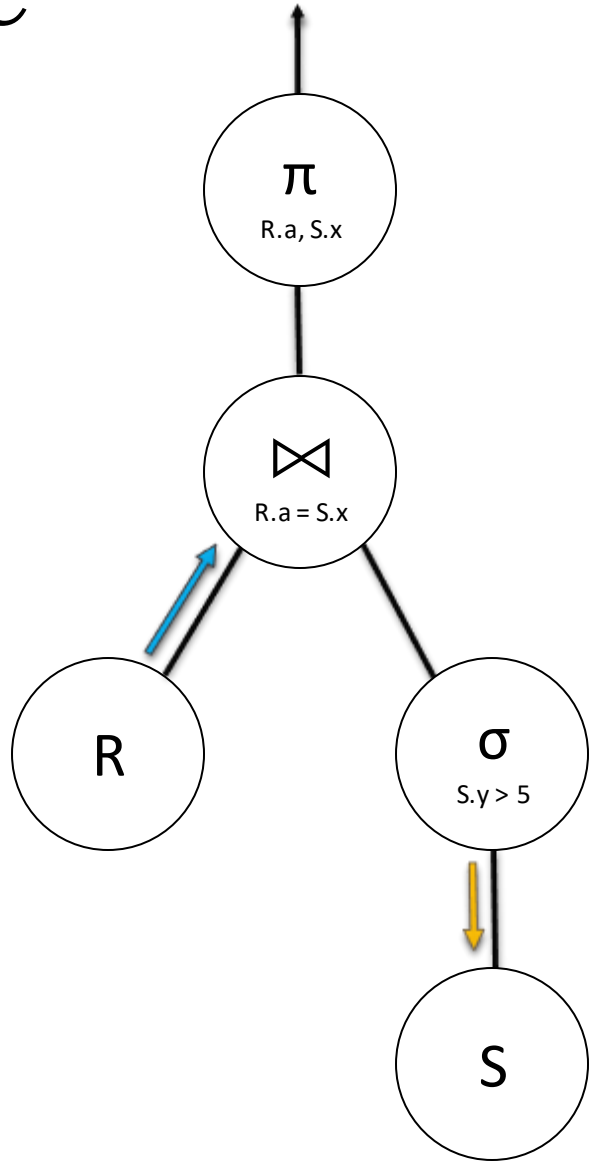
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

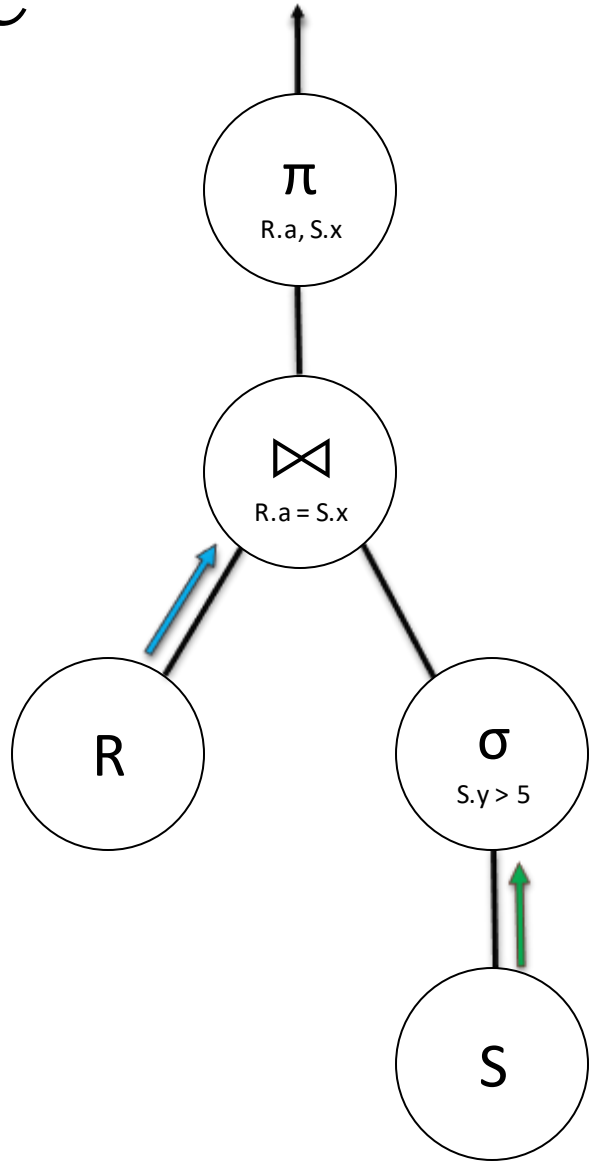
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

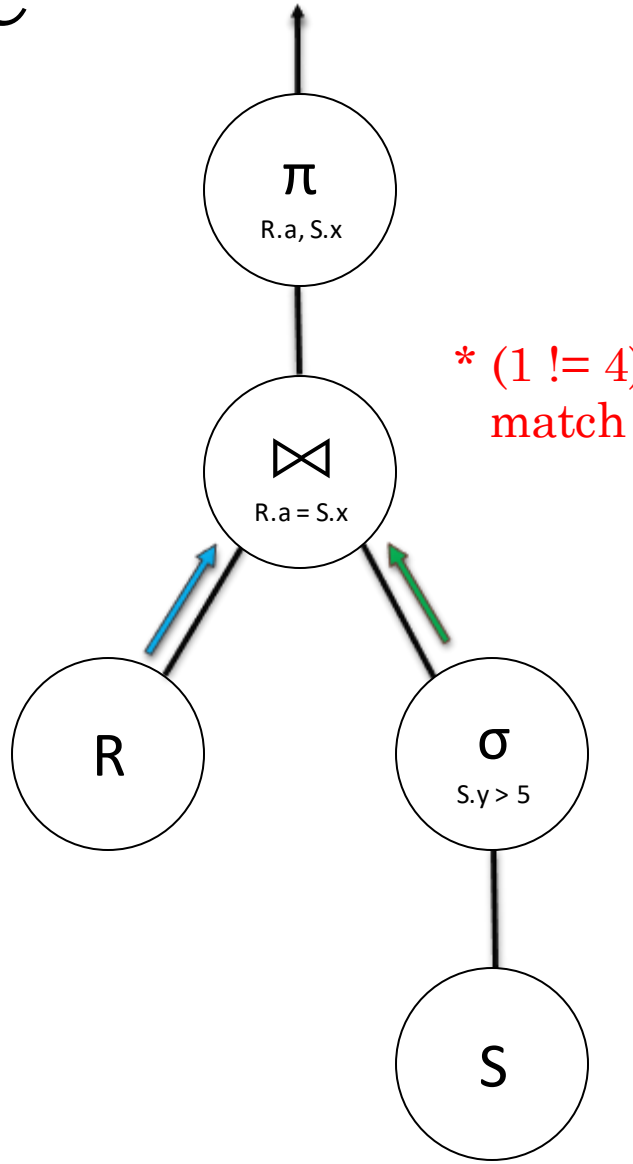
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



* (1 != 4) Keys do not match

Nested Loop Join Example

- **SQL**

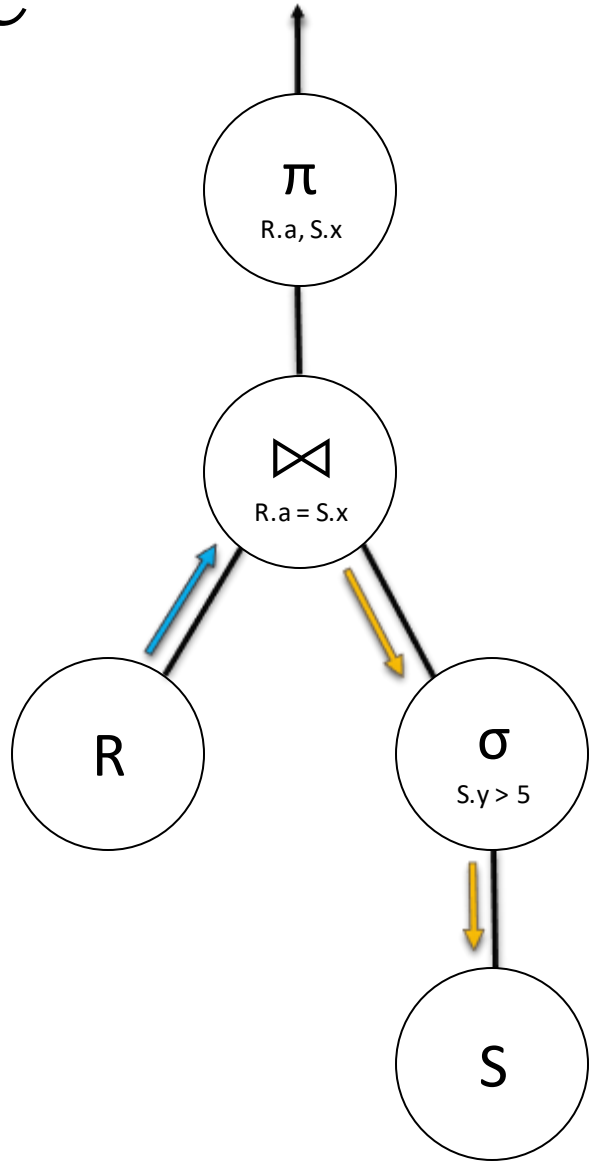
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Nested Loop Join Example

- **SQL**

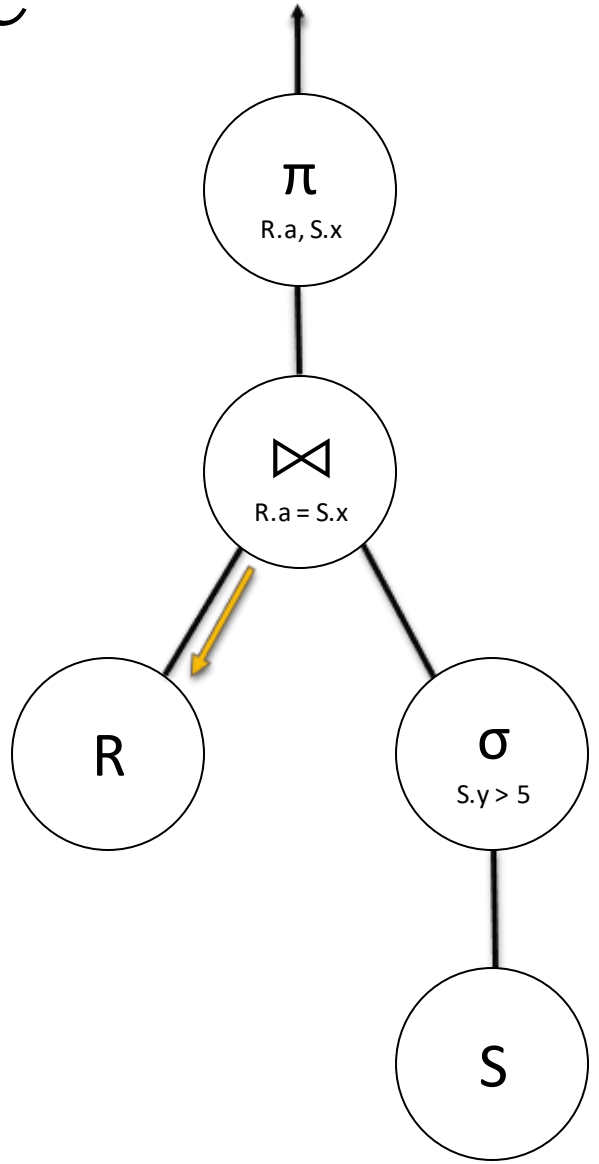
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



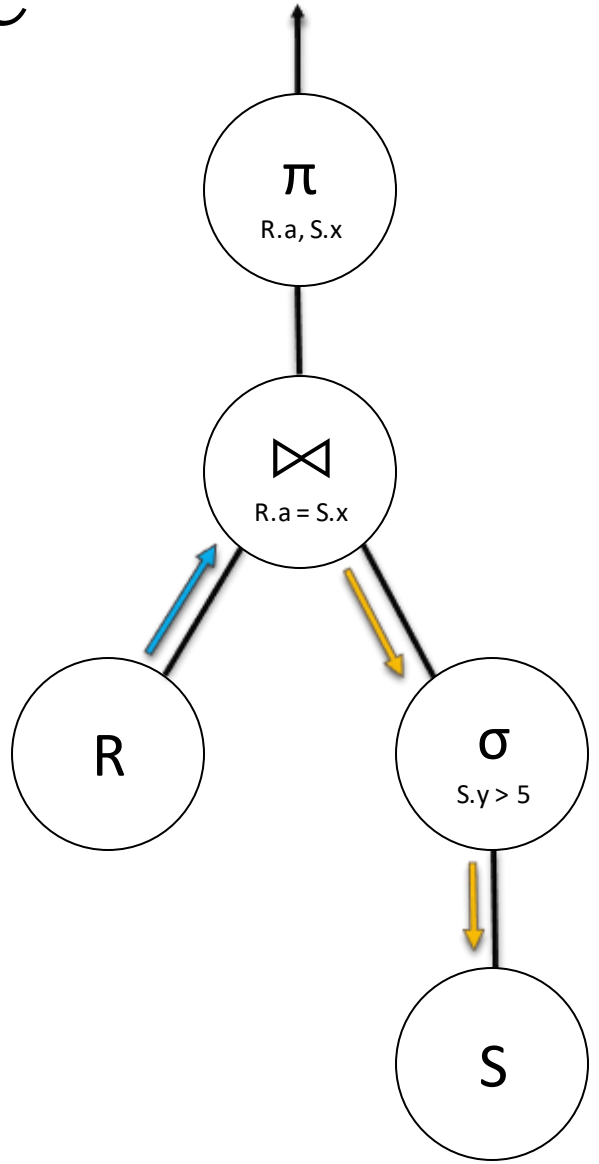
Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$



R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

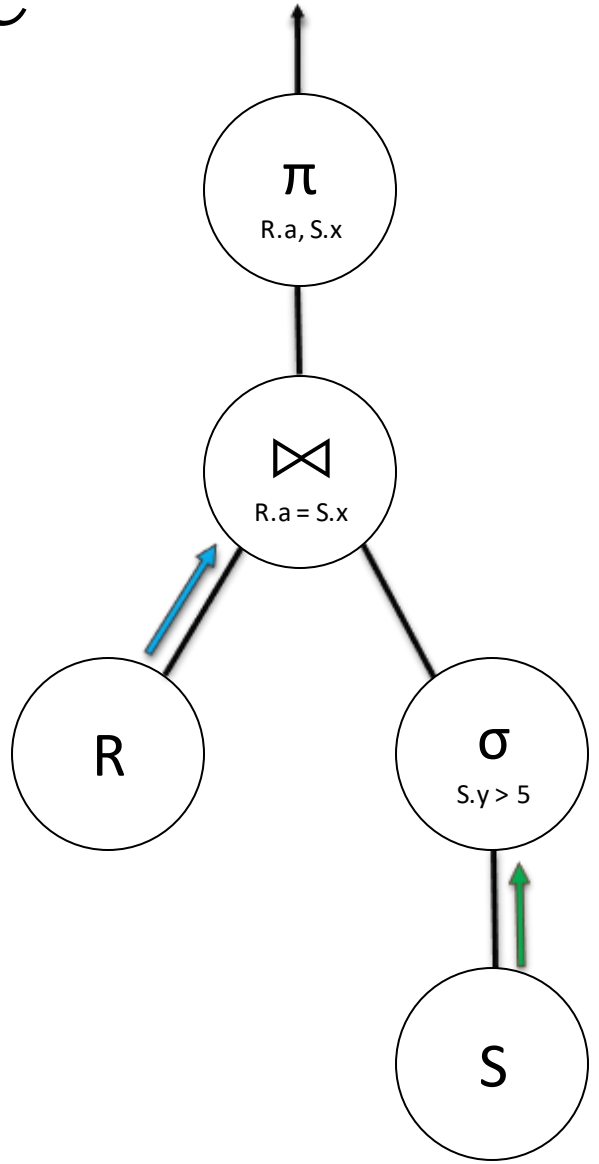
Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$



R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

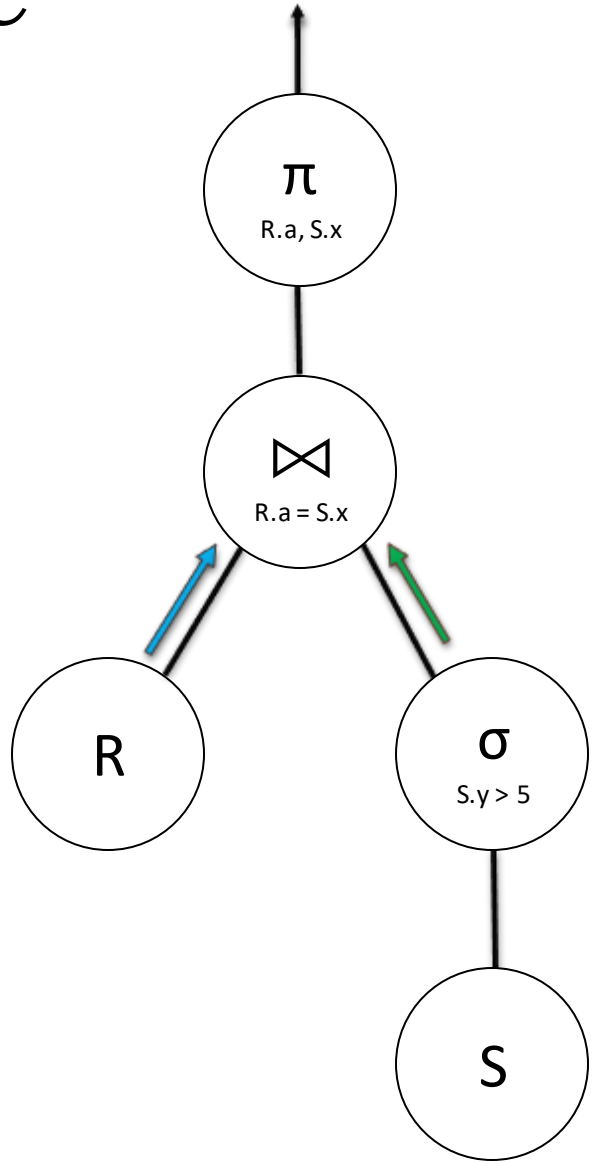
Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$



R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Nested Loop Join Example

- **SQL**

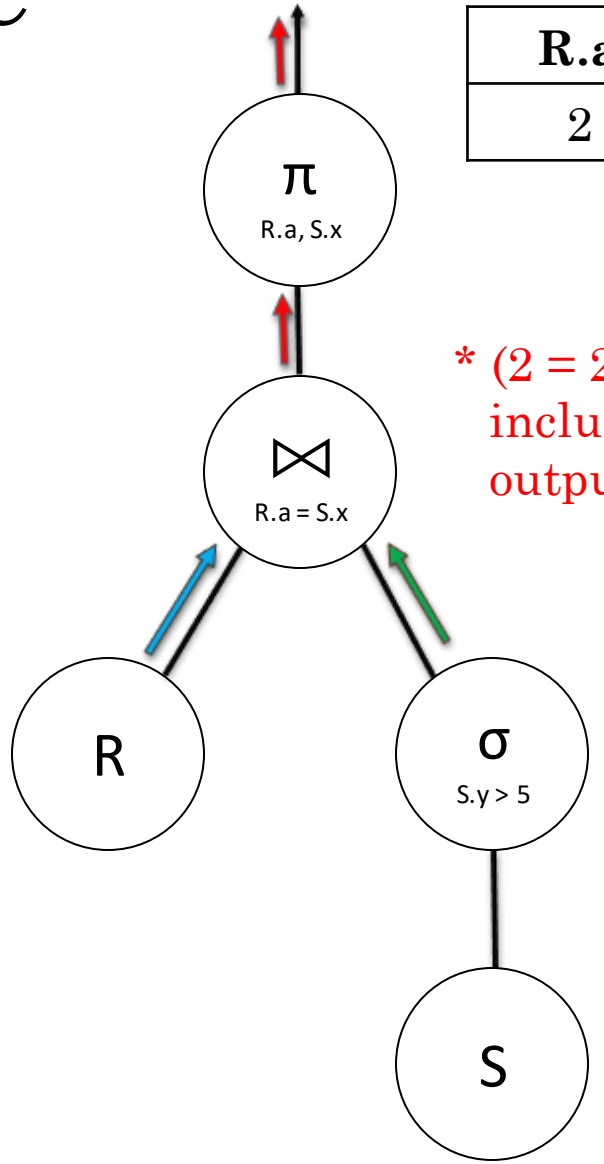
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

* (2 = 2) Keys match, include the tuple in output

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

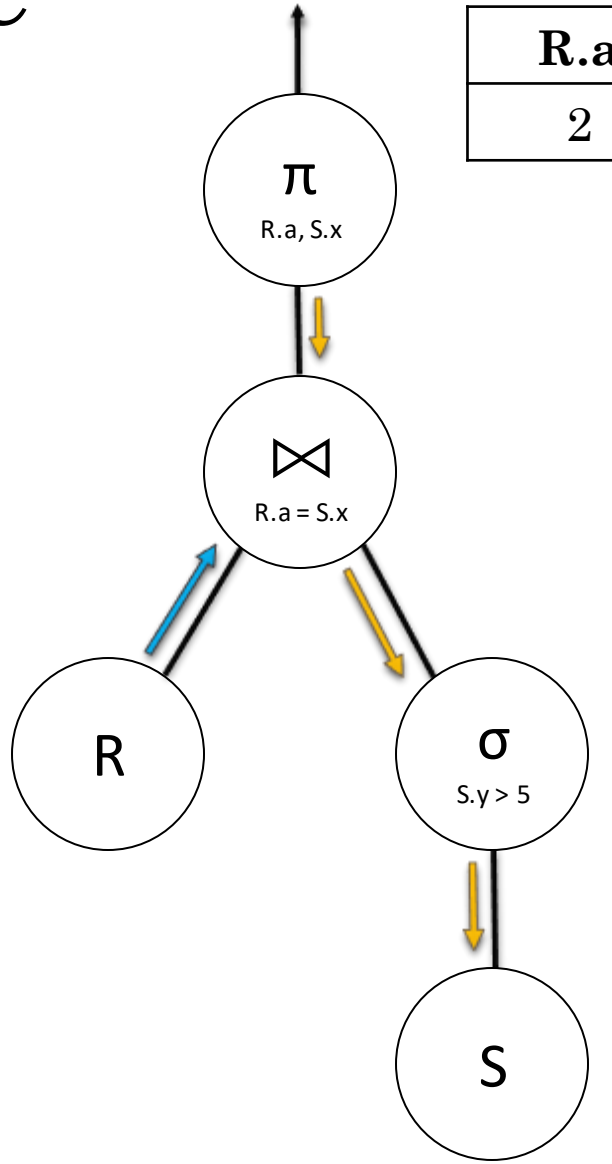
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



Nested Loop Join Example

- **SQL**

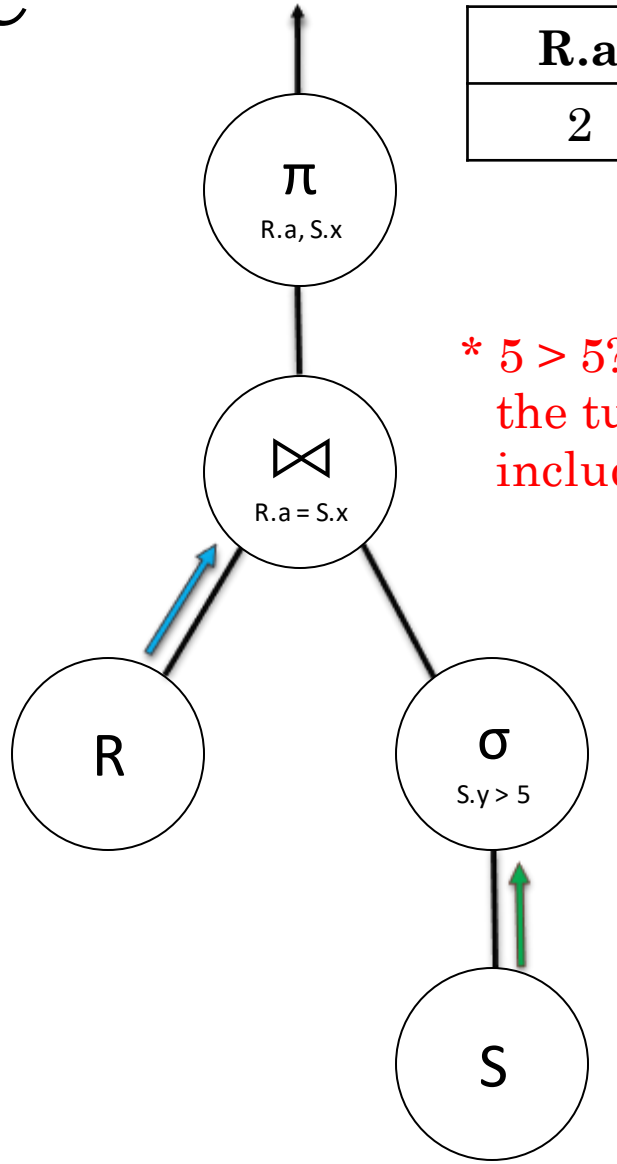
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

* 5 > 5? No, therefore the tuple is not included

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

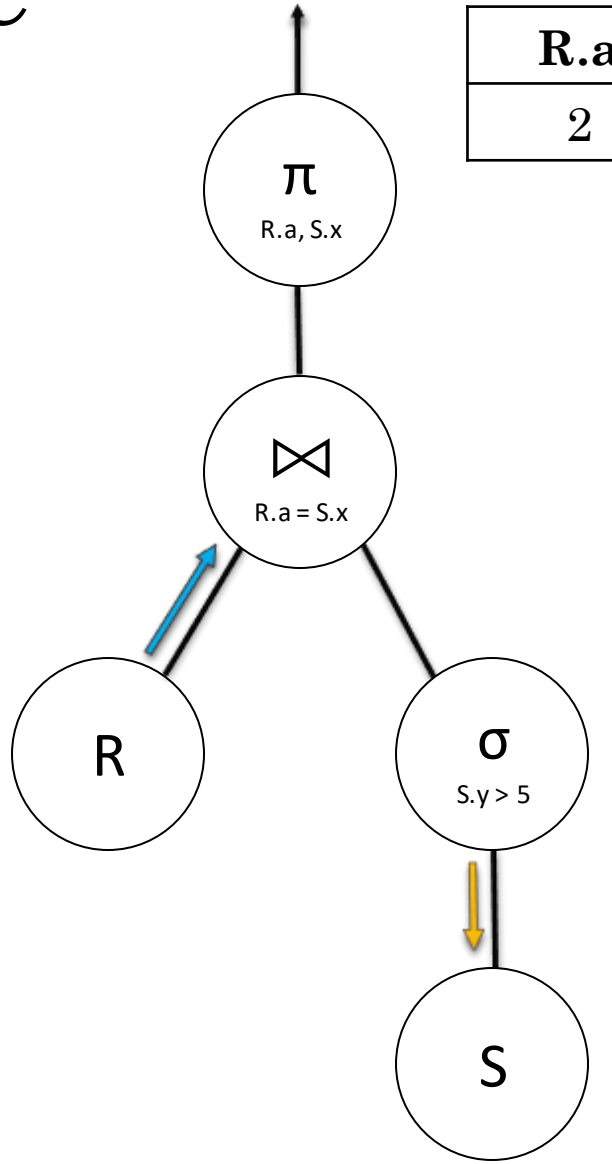
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



Nested Loop Join Example

- **SQL**

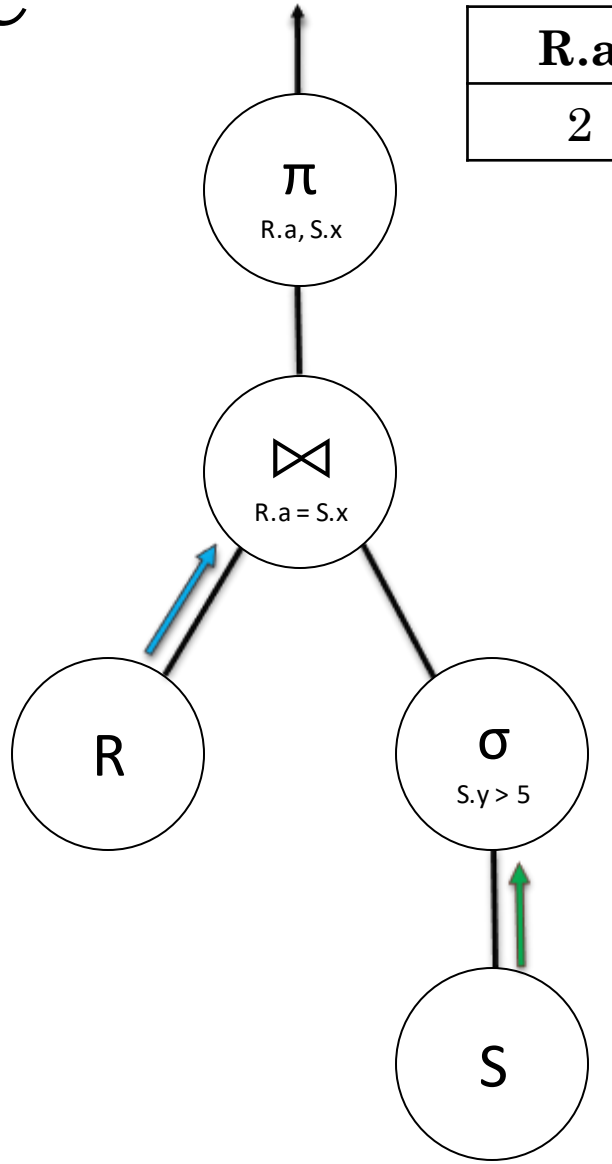
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

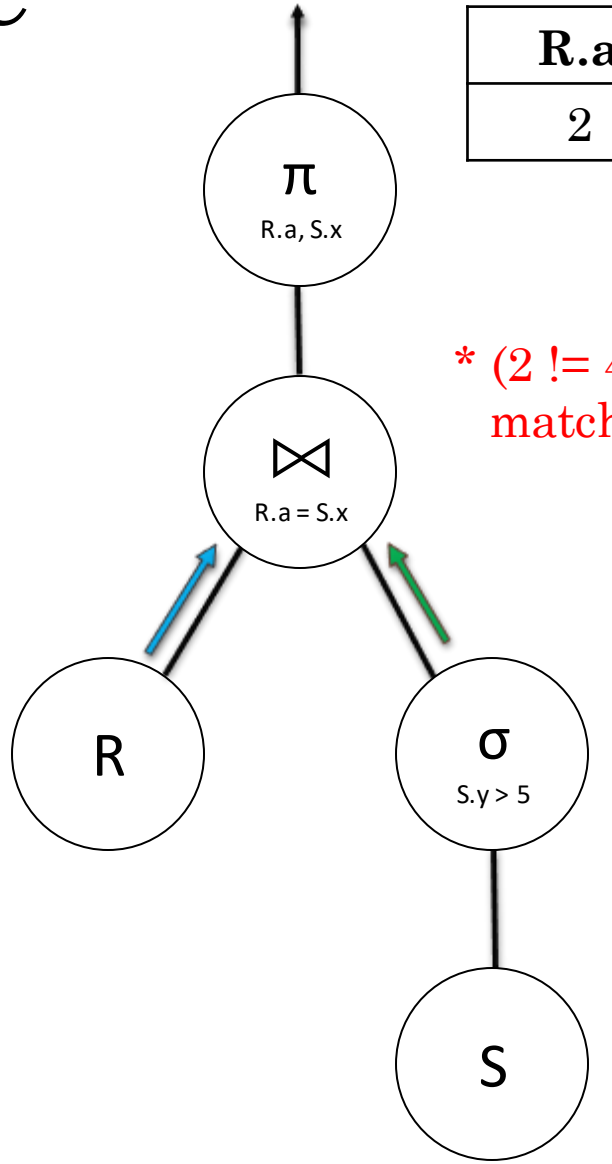
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



* (2 != 4) Keys do not match

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

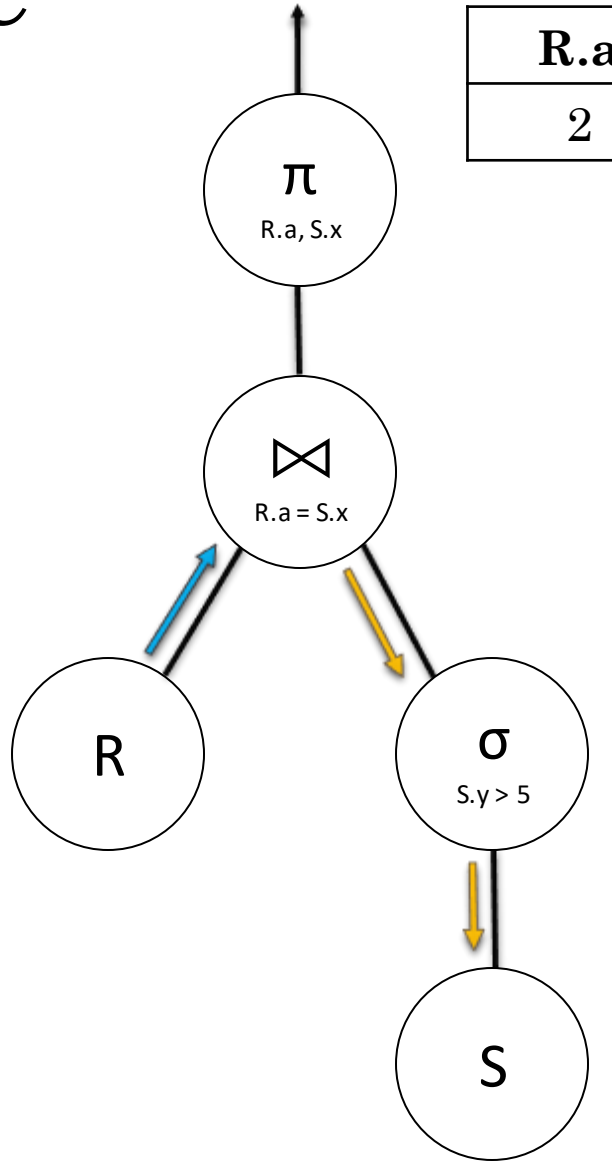
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

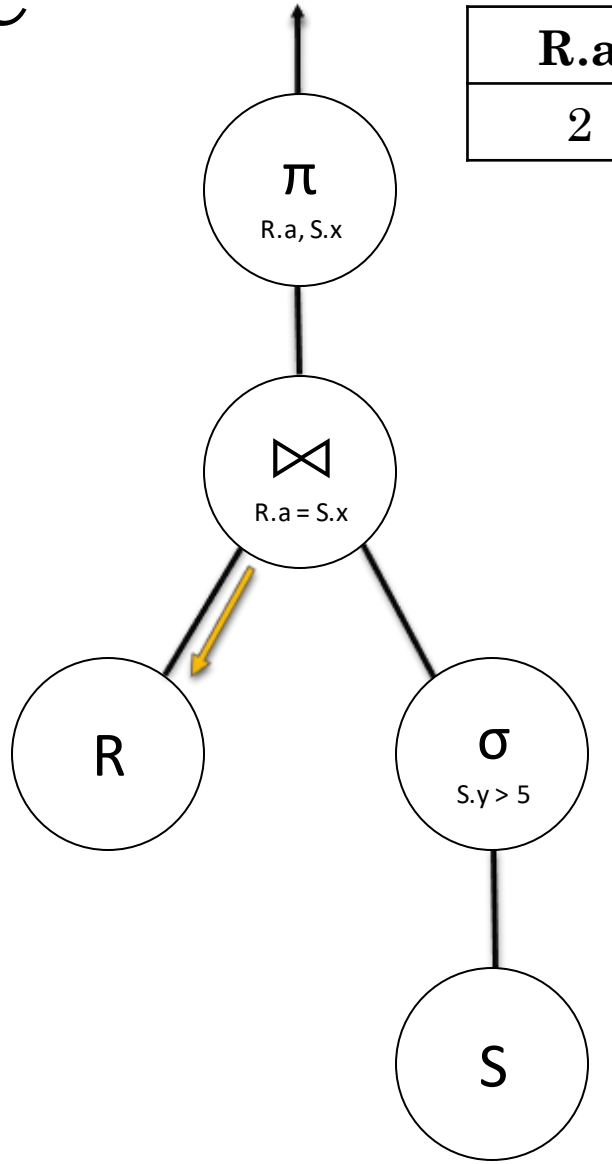
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

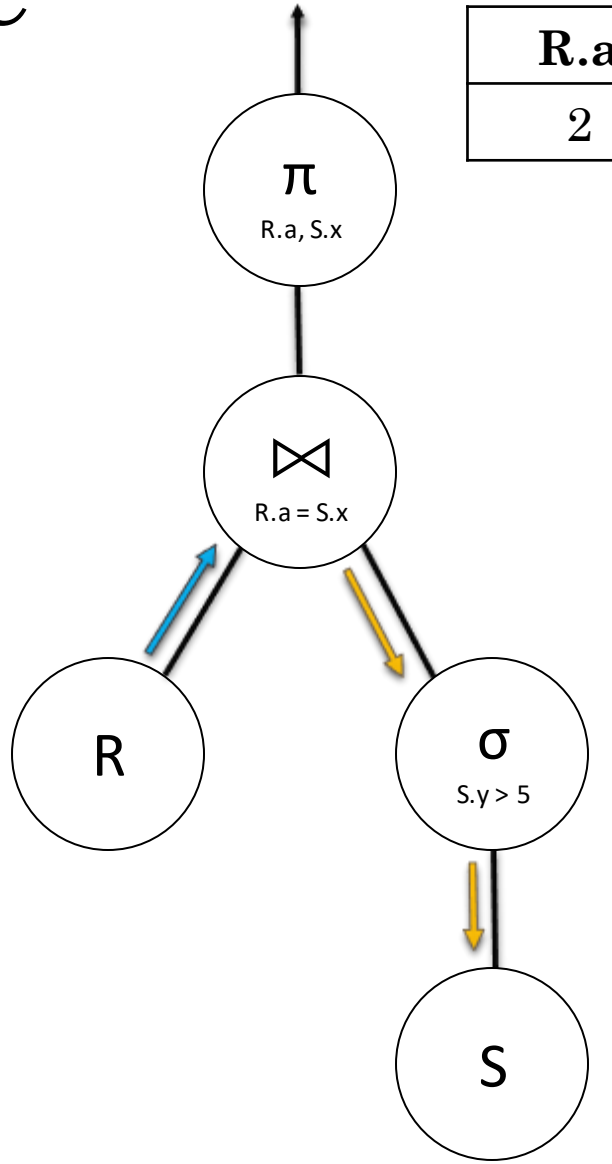
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



Nested Loop Join Example

- **SQL**

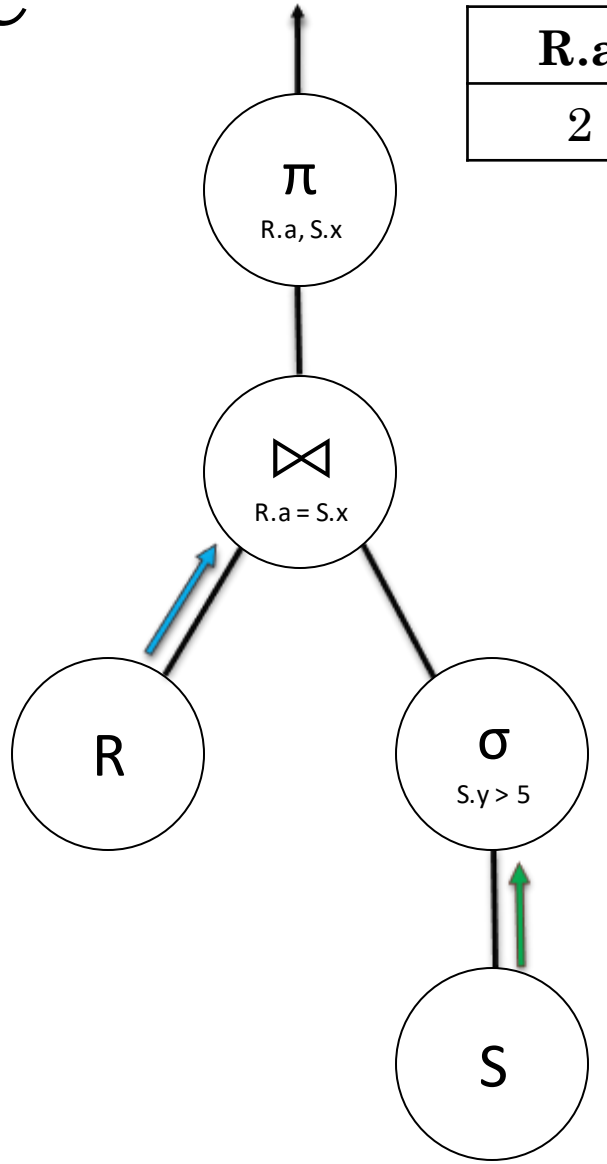
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

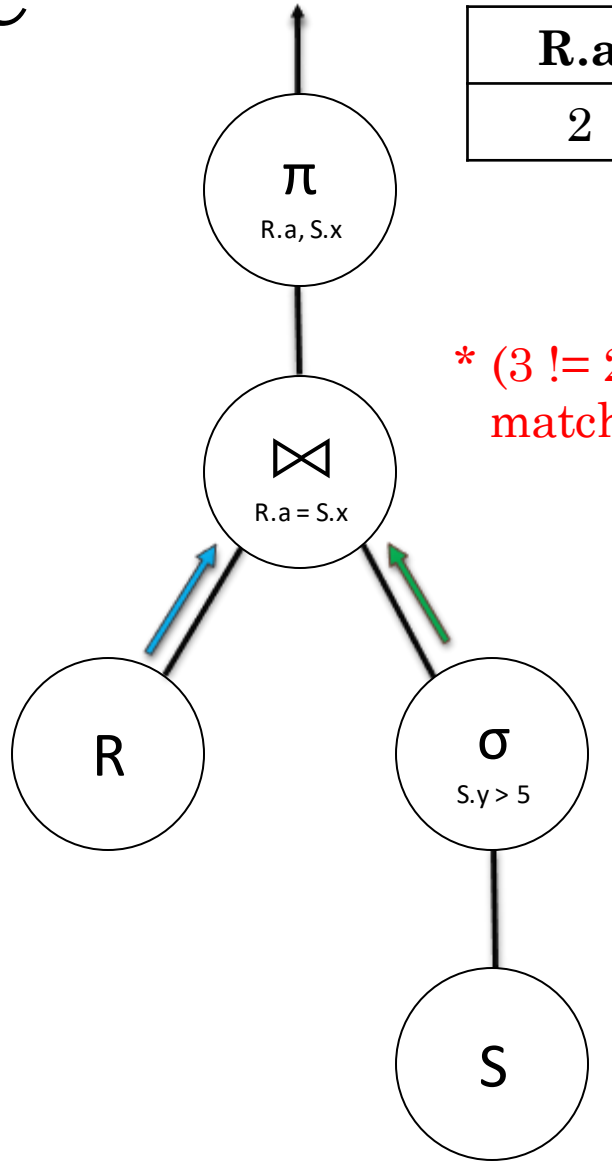
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



* (3 != 2) Keys do not match

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

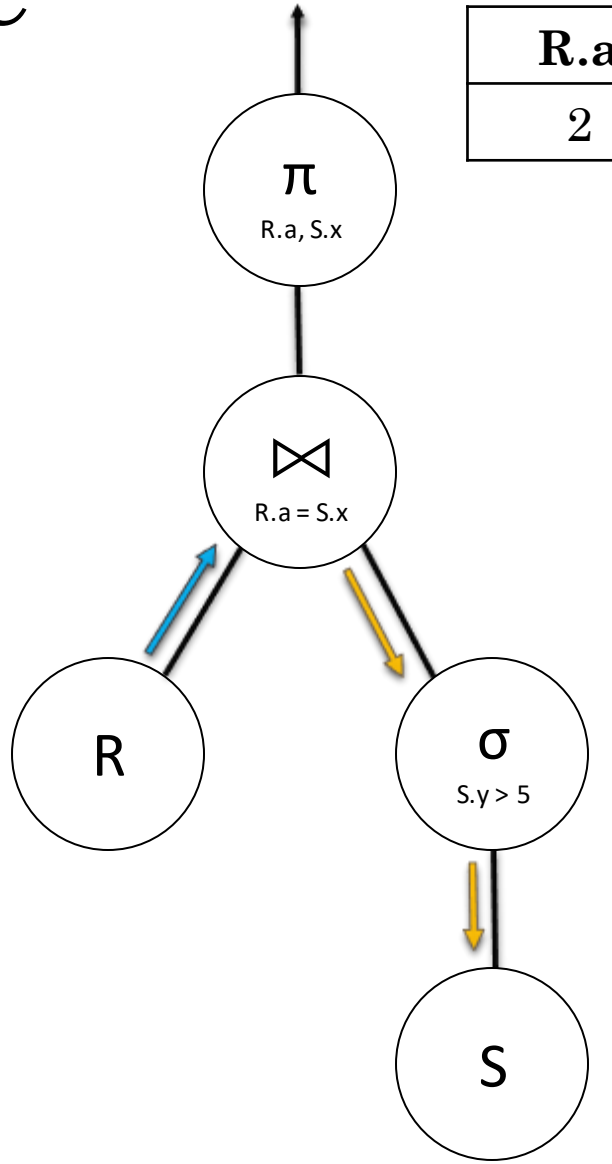
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



Nested Loop Join Example

- **SQL**

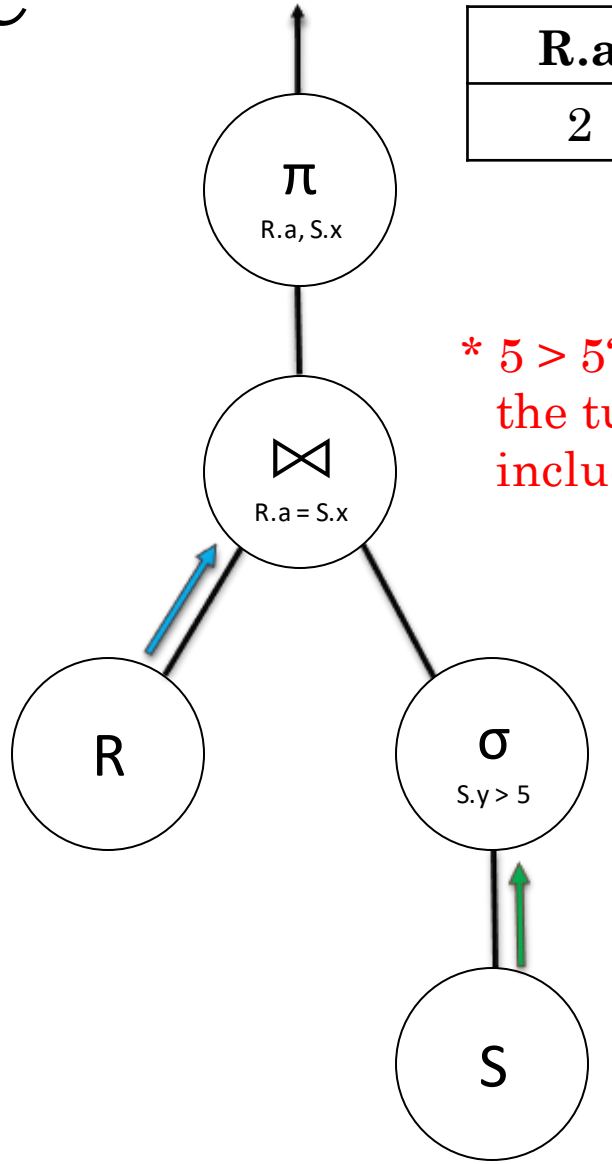
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

* 5 > 5? No, therefore the tuple is not included

Nested Loop Join Example

- **SQL**

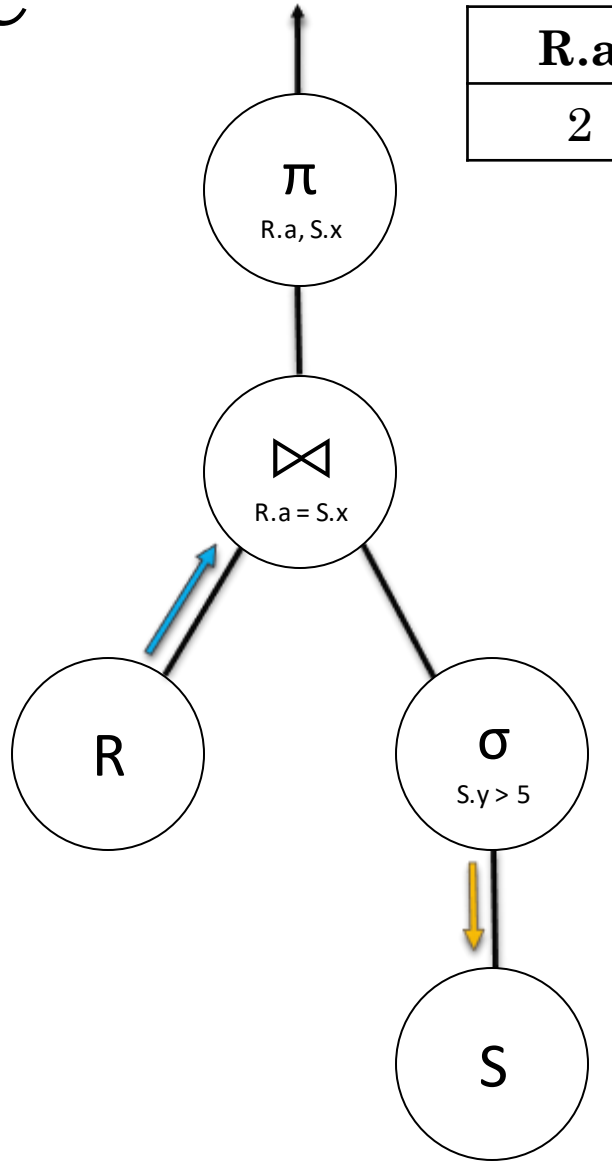
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

Nested Loop Join Example

- **SQL**

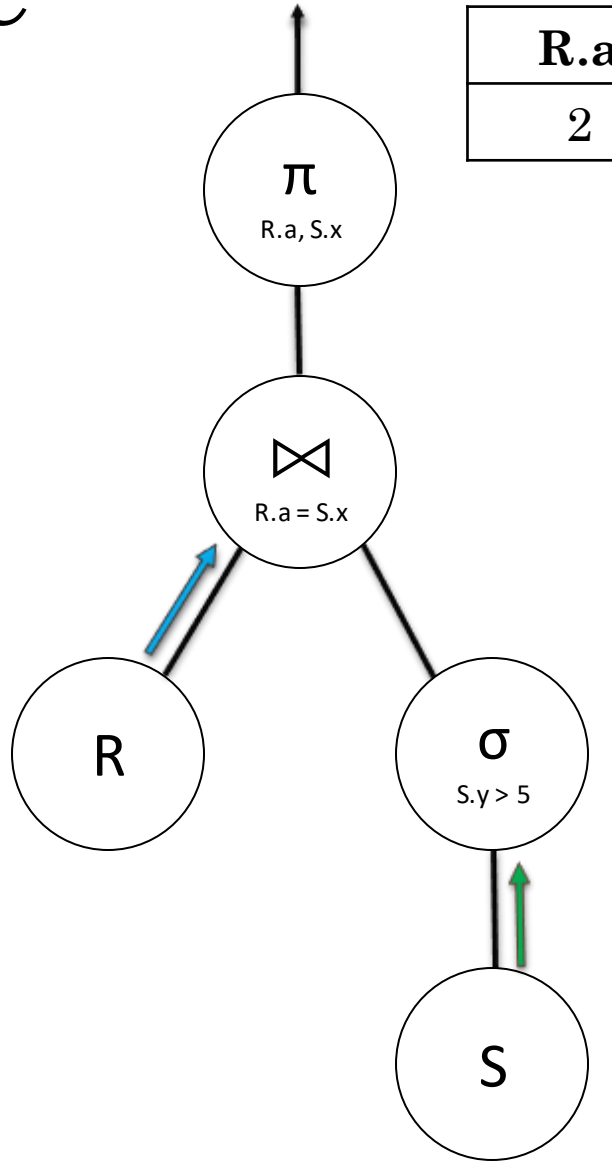
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

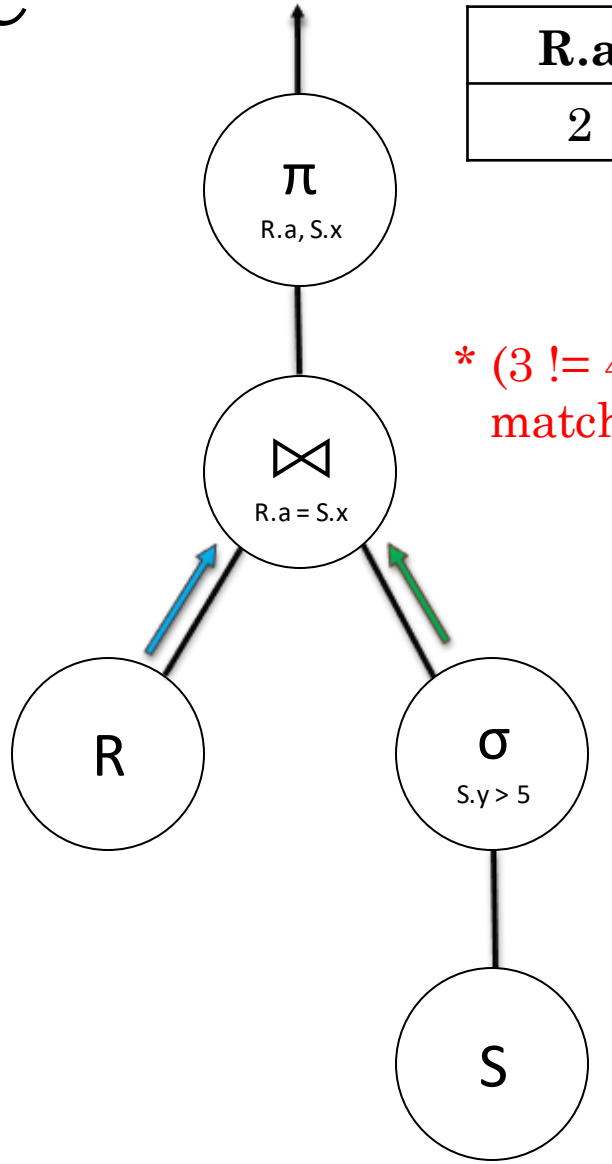
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



* (3 != 4) Keys do not match

Nested Loop Join Example

- **SQL**

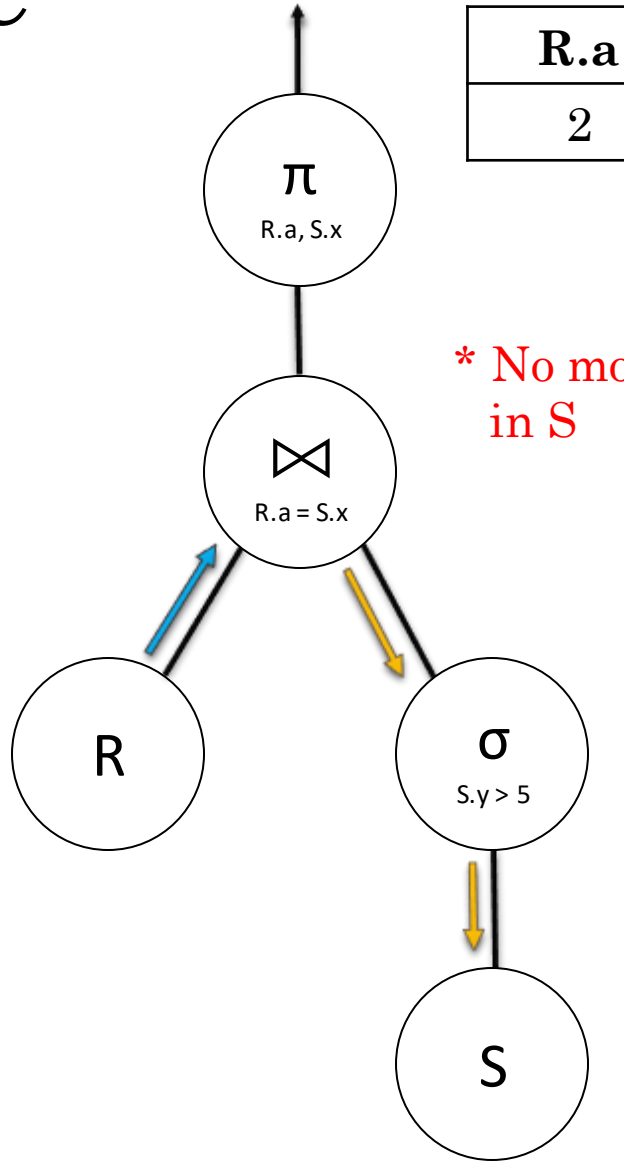
```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8



Output

R.a	S.X
2	2

* No more tuples left in S

Nested Loop Join Example

- **SQL**

```
SELECT R.a, S.x
  FROM R, S
 WHERE R.a = S.x AND S.y > 5;
```

- **Relational Algebra**

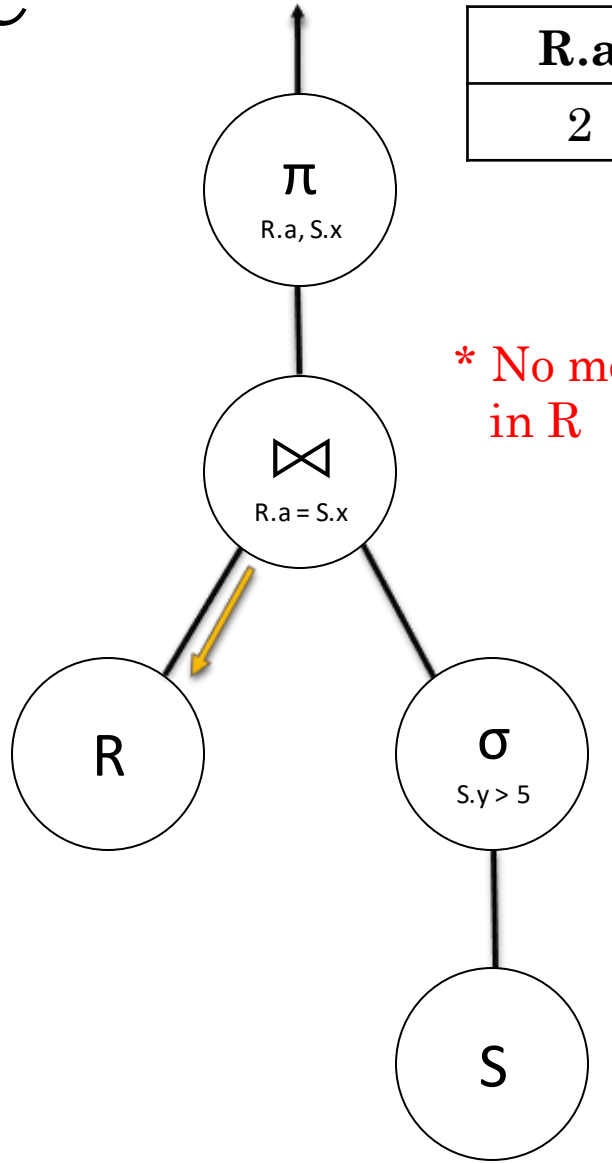
$\pi_{R.a,\ S.x}(\sigma_{S.y\ >\ 5}(S)\ \bowtie_{R.a\ =\ S.x} R)$

R.a	R.b
1	2
2	7
3	4

S.x	S.y
2	10
3	5
4	8

Output

R.a	S.X
2	2



* No more tuples left in R

Hash Join

- Hash Join is a specialized Join algorithm that only works when the join predicate is an equality predicate (e.g. R.a = S.x)
- Hash Join relies on *hashing*, and thus only equality is guaranteed to be a meaningful predicate for the hash function to operate on
- Hash Join only requires a *single pass through the input relations*, compared to a Nested Loop which requires iterating through multiple times

Implementing Hash Join in Minibase

- For *Project 3*, you will not have to implement a Hash Join entirely from scratch.
- You can rely on the already-existing *HashIndex* class and the *IndexScan class you will be implementing*.

Minibase Hash Join

1. Transform any input Iterators into an *IndexScan* on a *HashIndex* with the appropriate *SearchKey* for the join predicate
   - **You cannot simply cast the input Iterators as an IndexScan** – *you must build a temporary HeapFile and HashIndex and populate it with the result, and then open an IndexScan*.
   - You should now have two *IndexScan* operators, constructed on *HashIndexes* with the appropriate *SearchKey* – this means the buckets of the two HashIndexes should correspond.
2. Iterate through the *HashIndexes*, one bucket at a time. For each bucket, maintain a Hash Table of observed Tuples and report a match when it occurs
3. After exhausting all tuples in one bucket in both *IndexScans*, move to the next bucket. Repeat until all tuples are exhausted