

Implementing Relational Algebra Operators in a DBMS

Relational Algebra

- Operations that can be performed on sets to perform queries in a relational model
- Consist of:
 - 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
 - Many different algorithms exist for more efficient execution of a Join

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()`

Example (with Nested Loop Join)

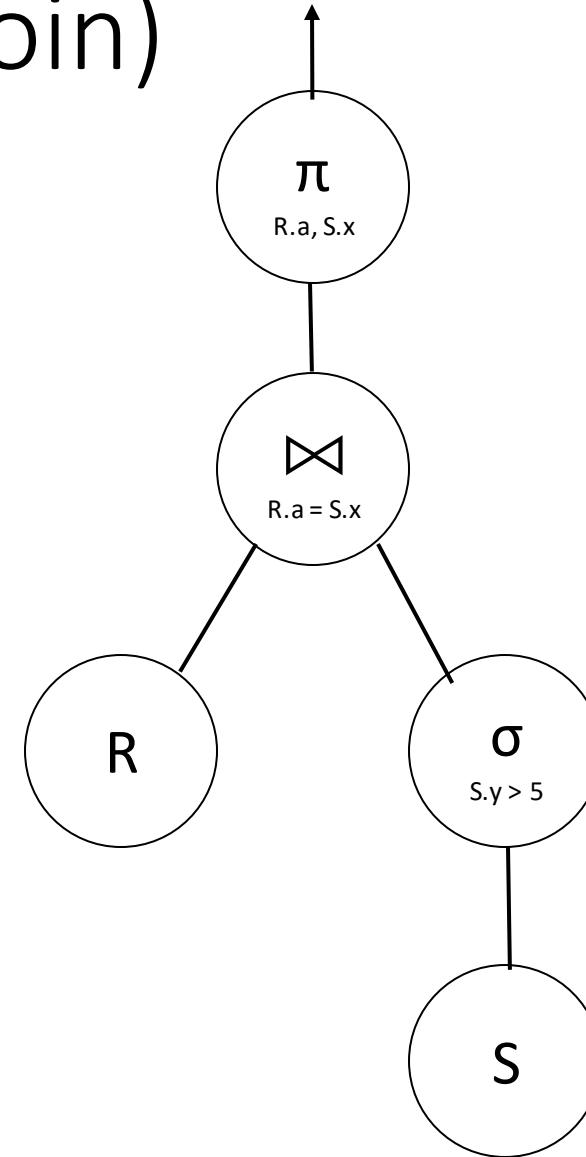
- SELECT R.a, S.x
FROM R, S
WHERE R.a = S.x AND S.y > 5;

- 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



Example

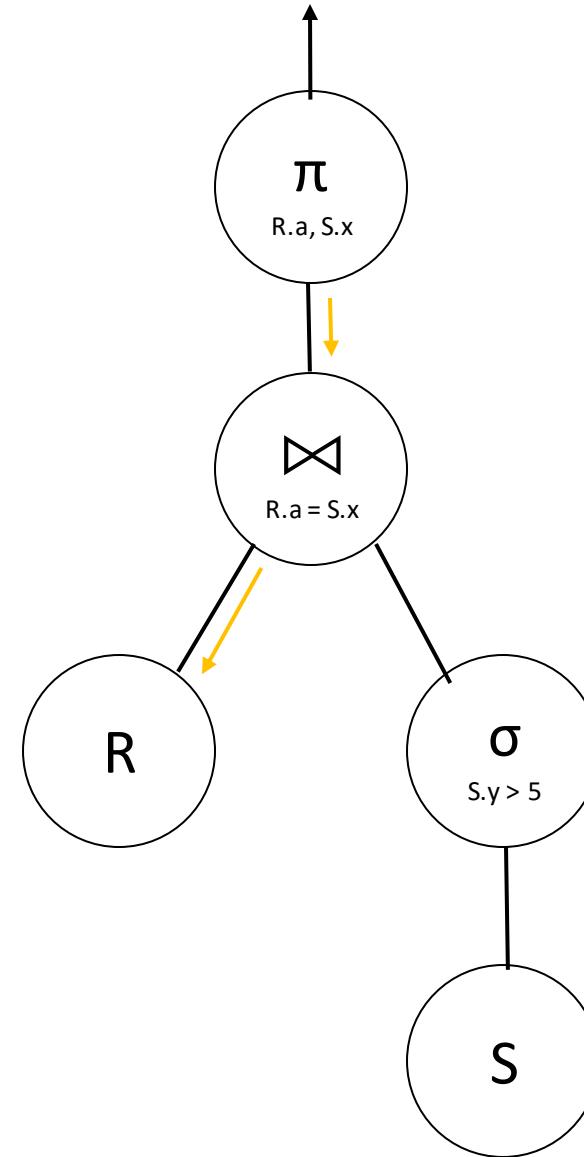
- SELECT R.a, S.x
FROM R, S
WHERE R.a = S.x AND S.y > 5;

- 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



Example

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

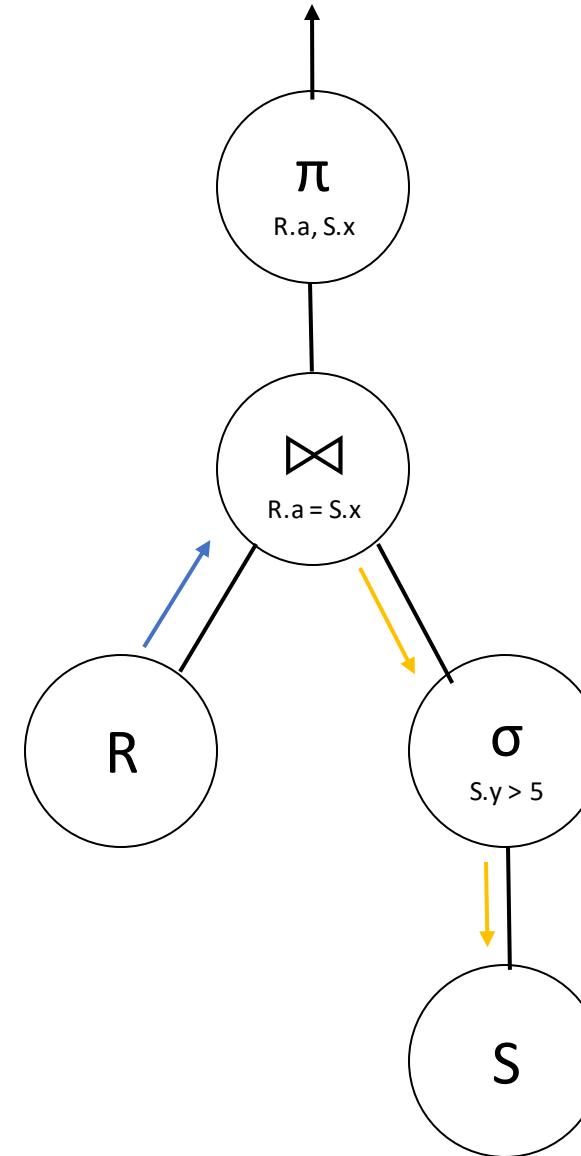
- Relational Algebra:

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



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

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

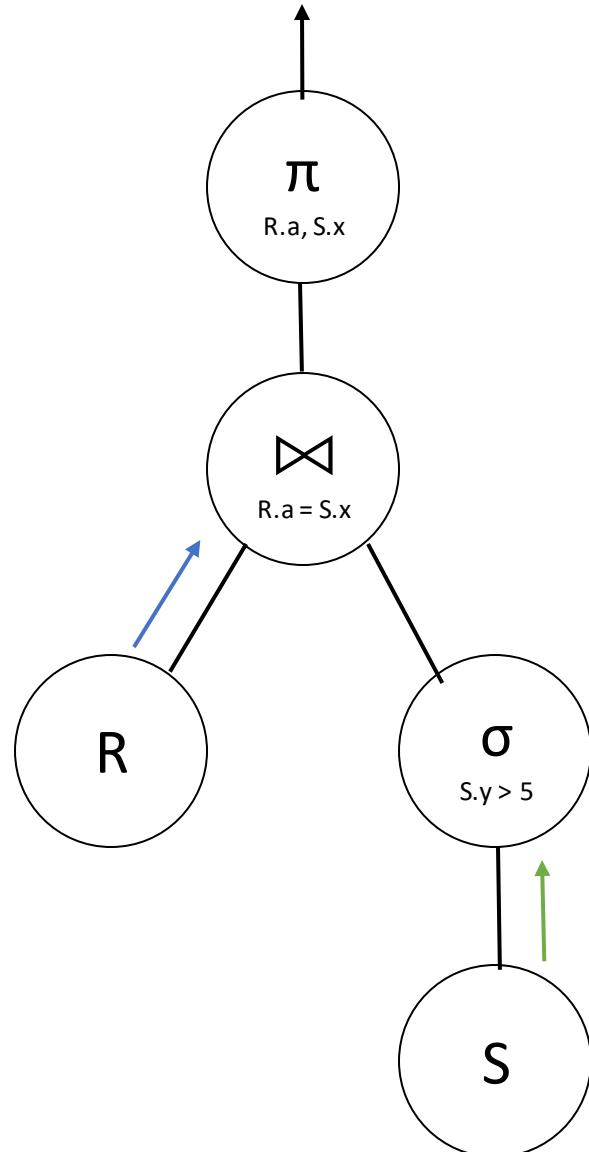


Example

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

- Relational Algebra:

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



A diagram illustrating the execution plan for the query. It shows a tree structure where the root node is a projection (π) with attributes $R.a, S.x$. Its children are a selection (σ) with condition $S.y > 5$ and a join (\bowtie) node. The selection node has a child relation R . The join node has two children: a selection node with condition $R.a = S.x$ and a relation S . A green double-headed arrow connects the S relation to its selection node, indicating they are joined together. A blue double-headed arrow connects the R relation to its selection node, indicating they are joined together. A green arrow points from the S relation up to the join node, and a blue arrow points from the R relation up to the join node.

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

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

Example

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

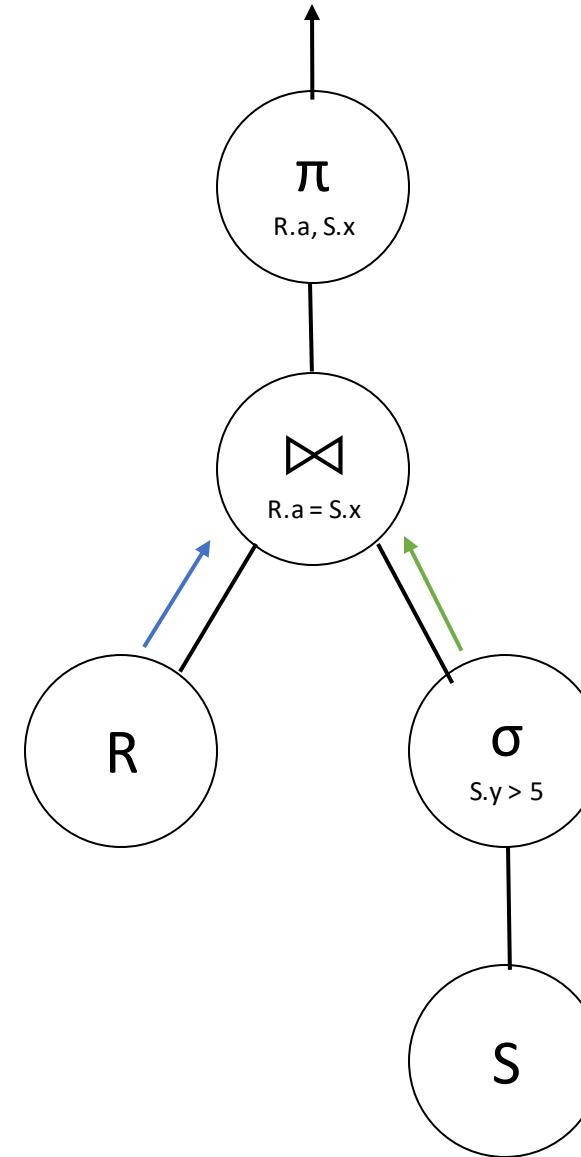
- Relational Algebra:

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

A flowchart illustrating the execution of the relational algebra query. It starts with two tables, R and S, represented by circles. A blue arrow points from table R to a projection node (π). A green arrow points from table S to a selection node (σ). The selection node has a condition $S.y > 5$. The output of the selection node and the input of the projection node both point to a join node (\bowtie). The join node has a condition $R.a = S.x$. The final output of the join node is the projection node, which outputs the selected attributes $R.a, S.x$.

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

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



Example

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

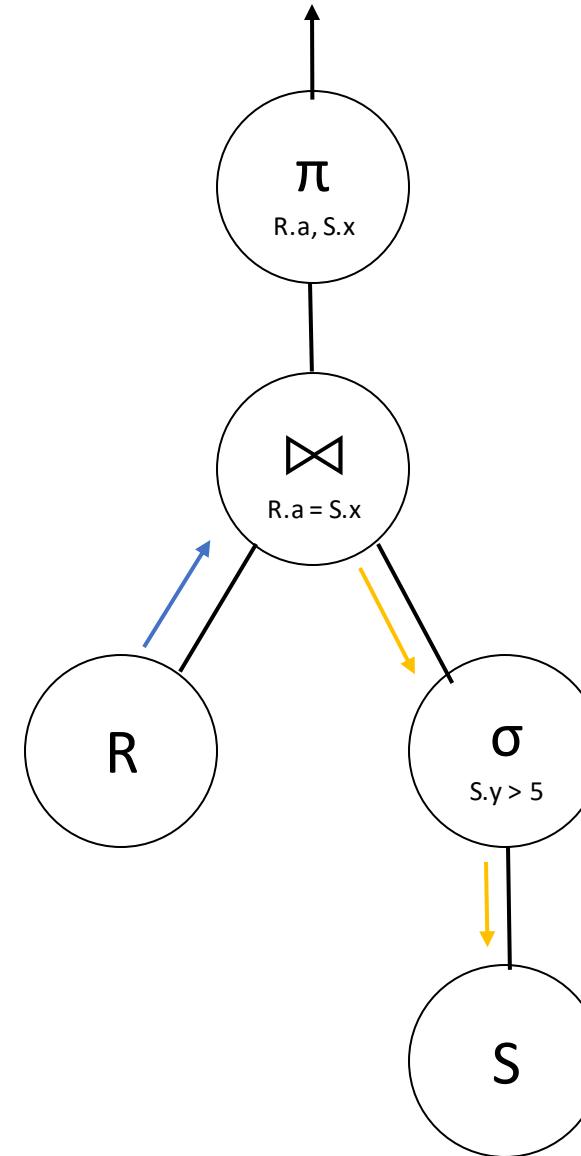
- Relational Algebra:

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

The diagram illustrates the flow of data through relational algebra operations. It starts with two tables, R and S, represented by circles. A blue arrow points from table R to a projection operation (π). A green arrow points from table S to a selection operation (σ). The projection operation has a label "R.a, S.x" and the selection operation has a label "S.y > 5". A black arrow connects the output of the projection to the input of the selection. Another black arrow connects the output of the selection to the final projection operation, which has a label "R.a, S.x".

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

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



Example

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

- Relational Algebra:

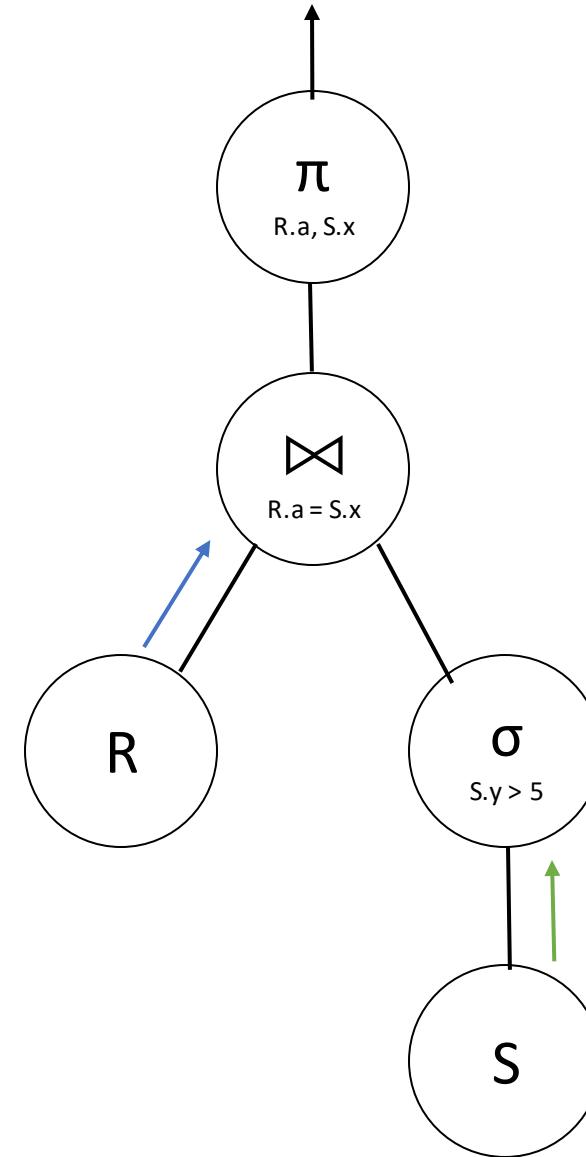
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$



A diagram illustrating the execution plan for the given query. It shows a tree of relational operations. At the bottom are two tables, R and S. Table R has columns R.a and R.b with rows (1, 2), (2, 7), and (3, 4). Table S has columns S.x and S.y with rows (2, 10), (3, 5), and (4, 8). A blue arrow points from the left side of table R towards the root node. A green arrow points from the right side of table S towards the sigma node. The root node is a projection operation $\pi_{R.a, S.x}$ with inputs R.a and S.x. Its children are a selection operation $\sigma_{R.a = S.x}$ and a join operation \bowtie . The selection node has input R.a and its child is table R. The join node has inputs R.a and S.x, with children table R and a sigma node $\sigma_{S.y > 5}$. The sigma node has input S.y and its child is table S.

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

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

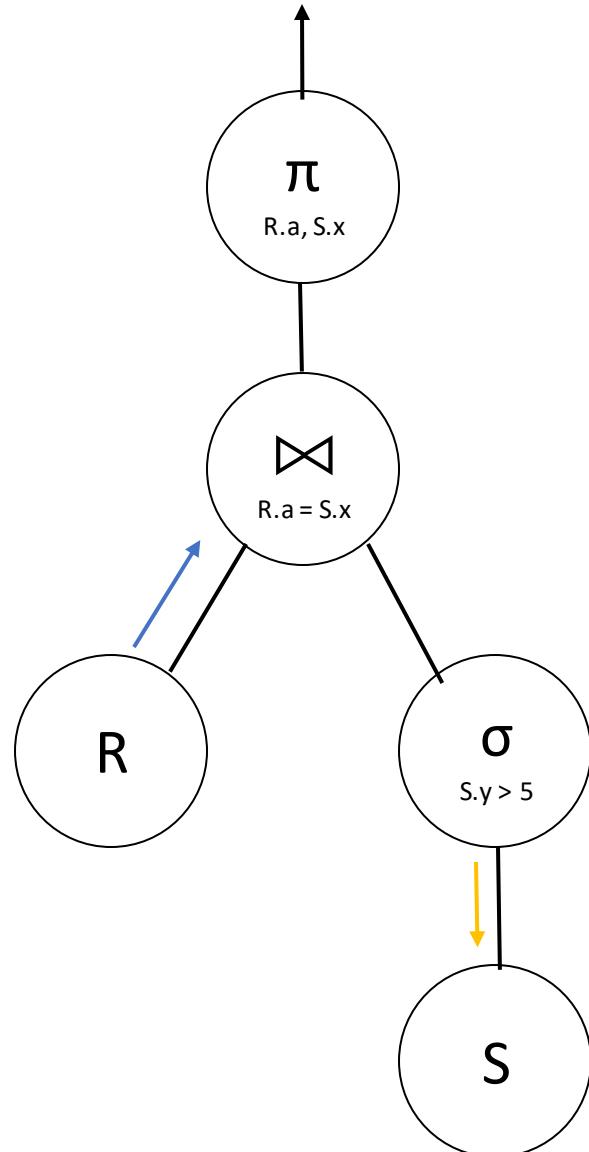


Example

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

- Relational Algebra:

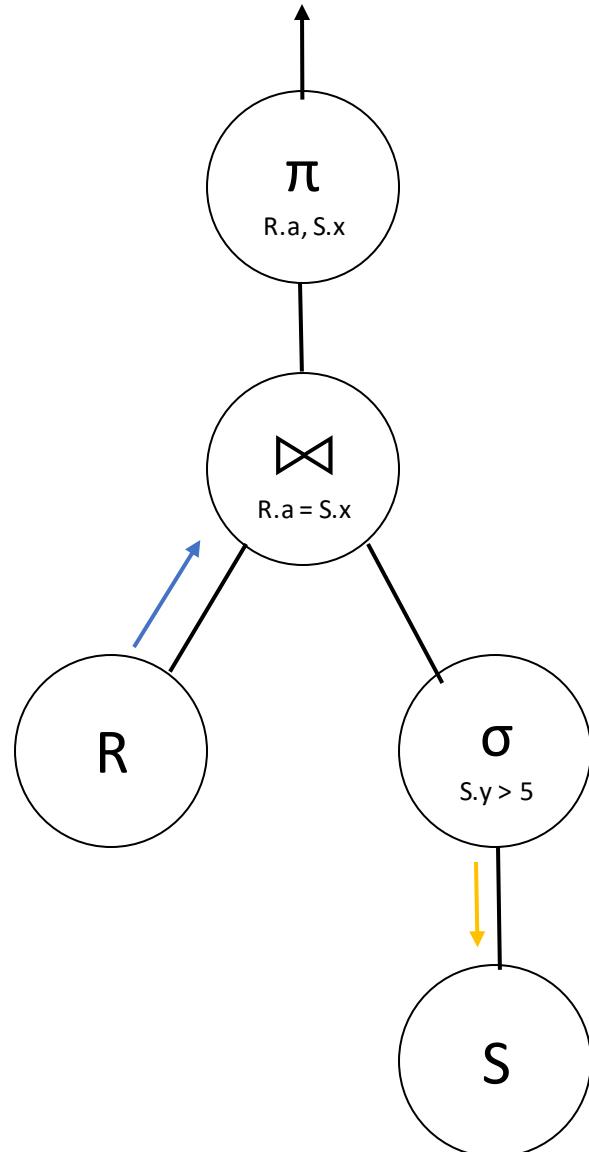
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$



A diagram illustrating the execution plan for the query. It shows a tree structure where the root node is a projection (π) with attributes $R.a, S.x$. Its children are a selection (σ) with condition $S.y > 5$ and a join (\bowtie) node. The selection node has a child relation R . The join node has two children: a selection node with condition $R.a = S.x$ and a relation S . A blue arrow points from the left to the relation R , and a green arrow points from the right to the relation S .

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

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

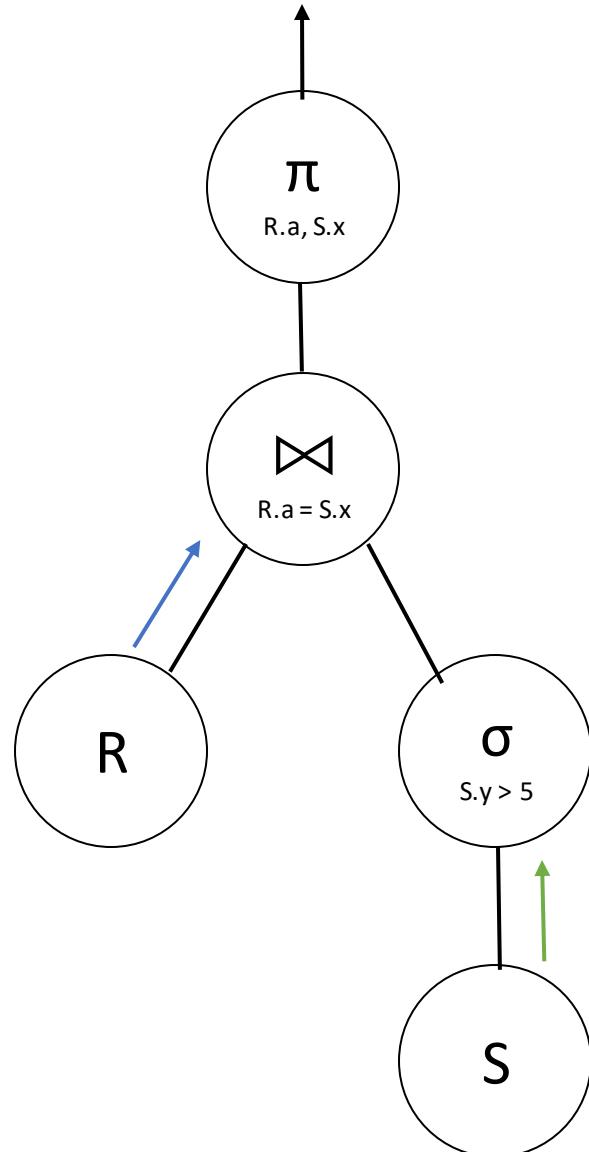


Example

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

- Relational Algebra:

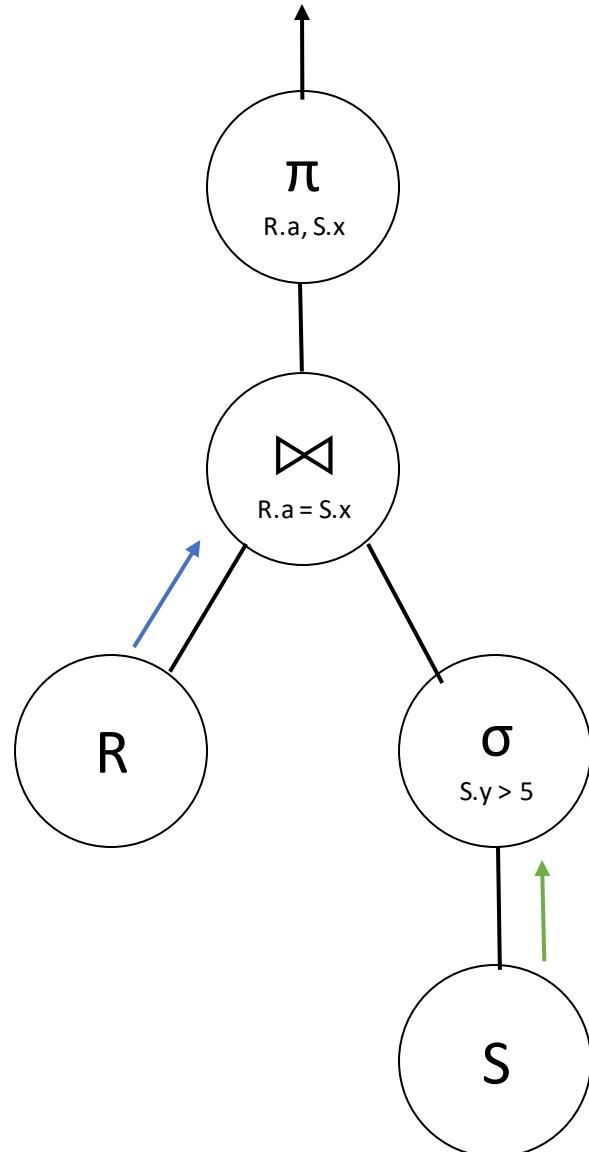
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$



A diagram illustrating the execution plan for the query. It shows a tree structure where the root node is a projection (π) with attributes $R.a, S.x$. Its children are a selection (σ) with condition $S.y > 5$ and a join (\bowtie) node. The selection node has a child relation R . The join node has two children: a selection node with condition $R.a = S.x$ and a relation S . A blue arrow points from the left to the relation R , and a green arrow points from the right to the relation S .

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

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



Example

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

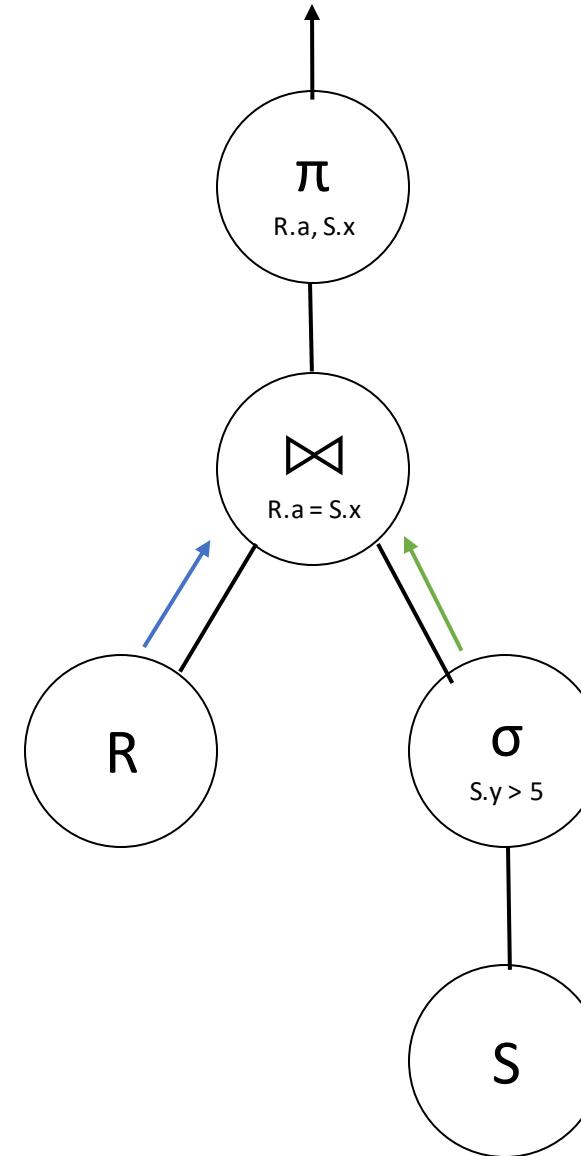
- Relational Algebra:

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

The diagram illustrates the execution plan for the relational algebra query. It starts with two tables, R and S, represented as rectangles. A blue arrow points from the left to table R, and a green arrow points from the right to table S. From table R, a blue arrow points up to a circle labeled σ with the condition $S.y > 5$. From table S, a green arrow points up to a circle labeled \bowtie with the condition $R.a = S.x$. Finally, a black arrow points up to a circle labeled π with the projection $R.a, S.x$.

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

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



Example

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

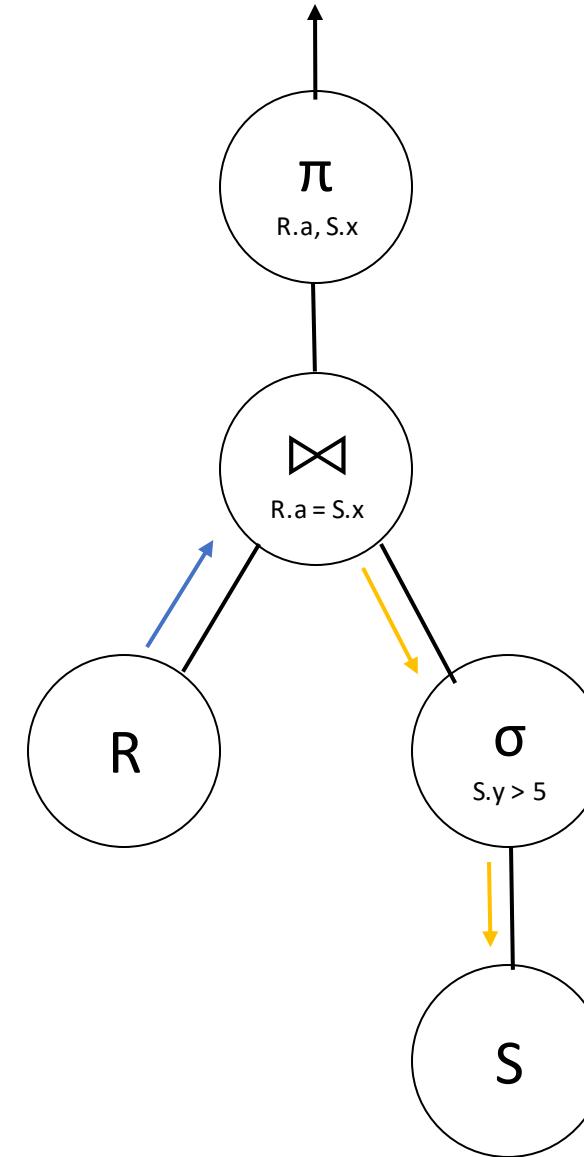
- Relational Algebra:

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

The diagram illustrates the execution plan for the relational algebra query. It starts with two relations, R and S, represented by circles. A blue arrow points to relation R, and a green arrow points to relation S. Relation R has columns R.a and R.b. Relation S has columns S.x and S.y. The query involves a selection operation σ_{S.y > 5} on relation S, represented by a circle with a sigma symbol. The result of this selection is then joined with relation R using the condition R.a = S.x, represented by a circle with a diamond symbol. Finally, the projection π_{R.a, S.x} is applied to the joined result, represented by a circle with a pi symbol.

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

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



Example

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

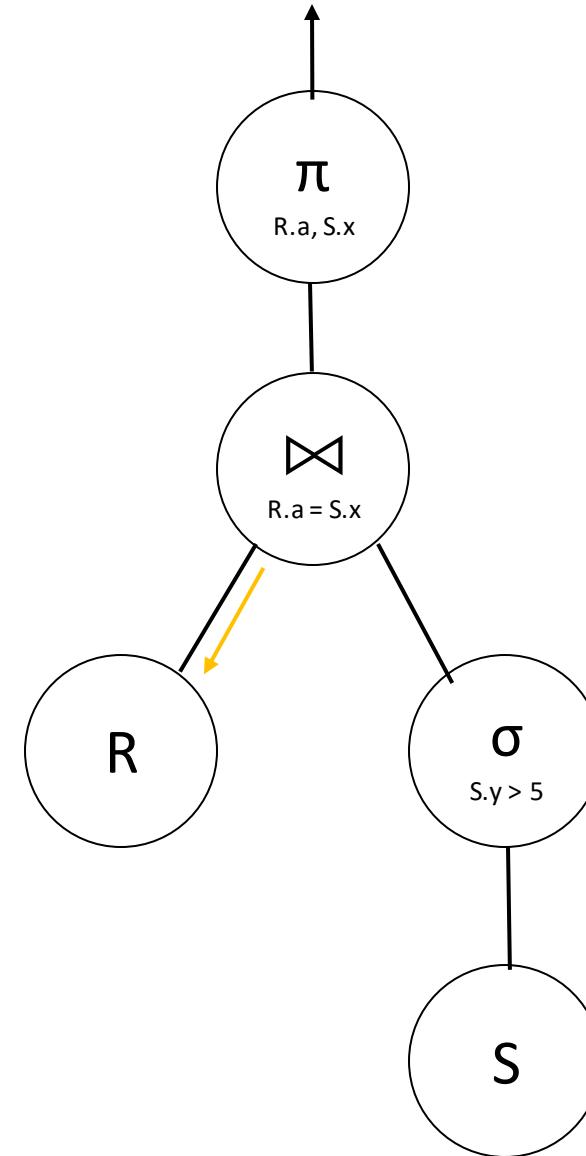
- Relational Algebra:

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



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

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



Example

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

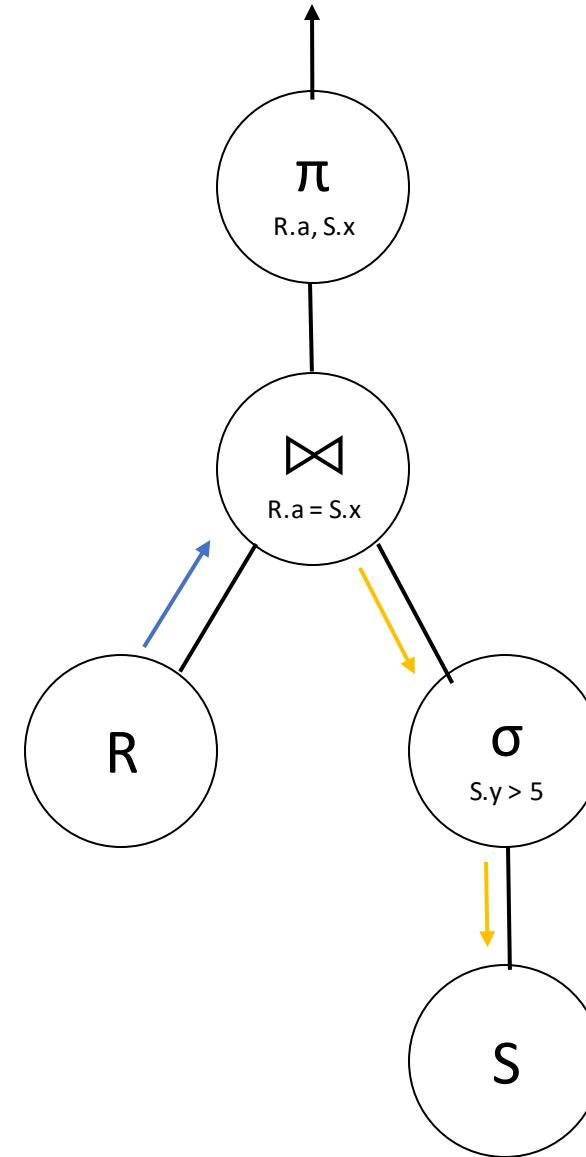
- Relational Algebra:

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



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

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

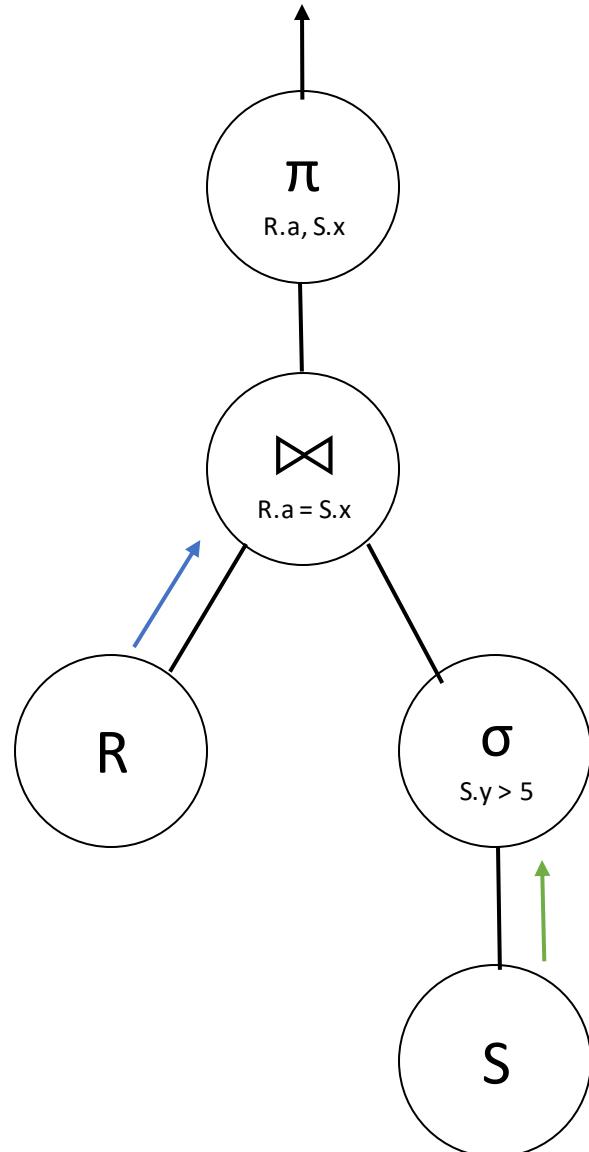


Example

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

- Relational Algebra:

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



A diagram illustrating the execution plan for the relational algebra query. It shows a tree structure where the root node is a projection (π) with attributes $R.a, S.x$. Its children are a selection (σ) with condition $S.y > 5$ and a join (\bowtie) node. The selection node has a child relation S . The join node has two children: a relation R and a selection (σ) with condition $R.a = S.x$. A blue arrow points from the relation R to the join node, and a green arrow points from the selection node $S.y > 5$ to its child node S . To the left of the tree, there are two tables: R (with columns $R.a$ and $R.b$) and S (with columns $S.x$ and $S.y$). A blue arrow points from the table R to the relation R , and a green arrow points from the table S to the selection node $S.y > 5$.

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

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

Example

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

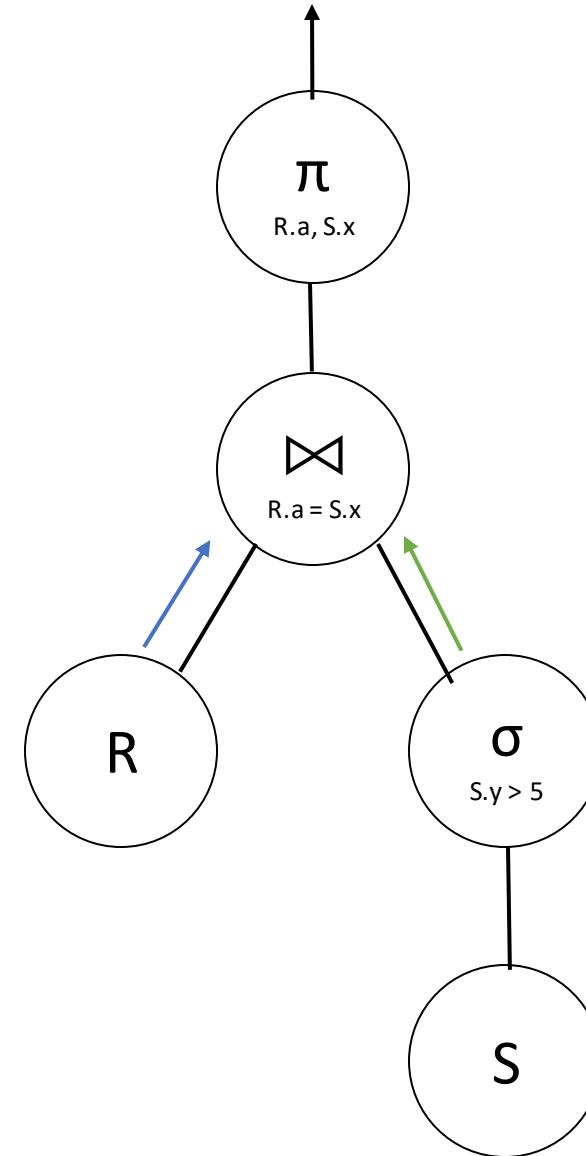
- Relational Algebra:

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

A flowchart illustrating the execution of the relational algebra query. It starts with two tables, R and S, represented by circles. A blue arrow points from the left towards the table R. A green arrow points from the right towards the table S. The table R has columns R.a and R.b with values 1, 2, 3 respectively. The table S has columns S.x and S.y with values 2, 10; 3, 5; 4, 8 respectively.

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

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



Example

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

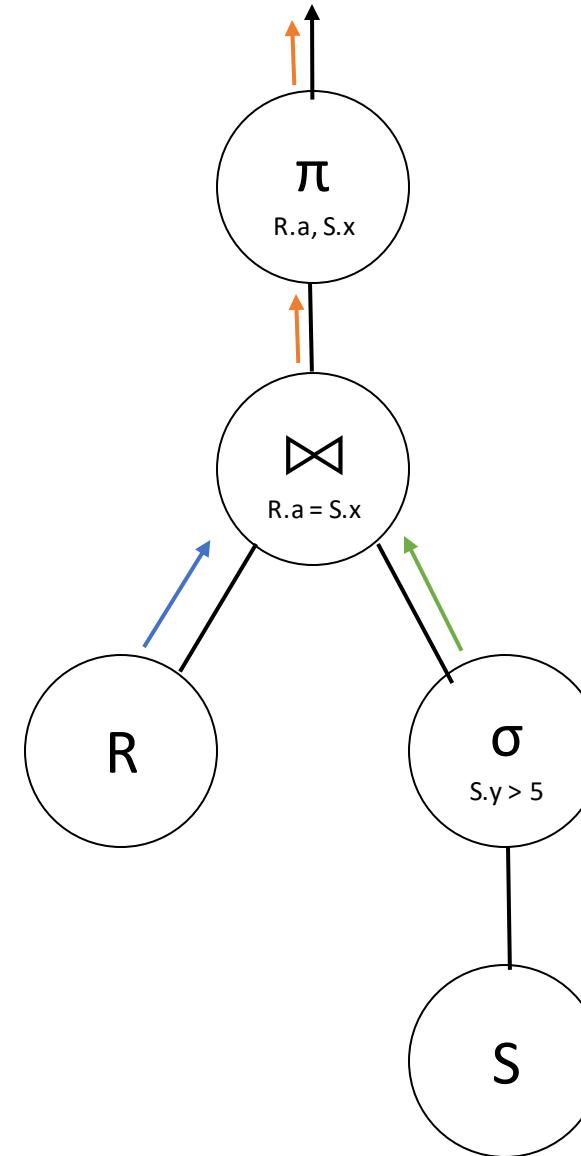
- Relational Algebra:
 $\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$



The diagram illustrates the execution plan for the relational algebra query. It starts with two relations, R and S, represented by circles. A blue arrow points from relation R to a selection operation circle labeled σ with the condition $S.y > 5$. A green arrow points from relation S to a join operation circle labeled \bowtie with the condition $R.a = S.x$. The output of the selection operation and the input of the join operation are connected by a black arrow. From the join operation, a blue arrow points to a projection operation circle labeled π with the attributes $R.a, S.x$. A green arrow points from the join operation to the projection operation. A final orange arrow points upwards from the projection operation, indicating the final result.

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

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



Example

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

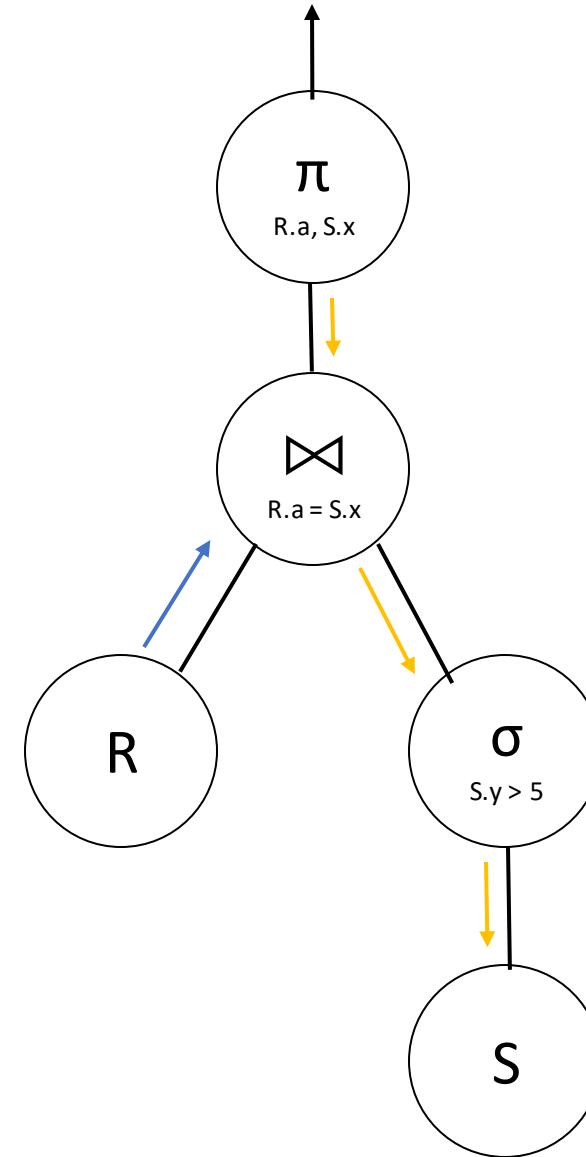
- Relational Algebra:

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

A flowchart illustrating the execution of the relational algebra query. It starts with two tables, R and S, represented by circles. A blue arrow points from R to the projection node π . A green arrow points from S to the selection node σ . The selection node σ has an output arrow pointing to the join node \bowtie . The join node \bowtie has two input arrows: one from R and one from the selection node σ . The join node \bowtie has an output arrow pointing to the projection node π . The projection node π has an output arrow pointing upwards.

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

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



Example

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

- Relational Algebra:

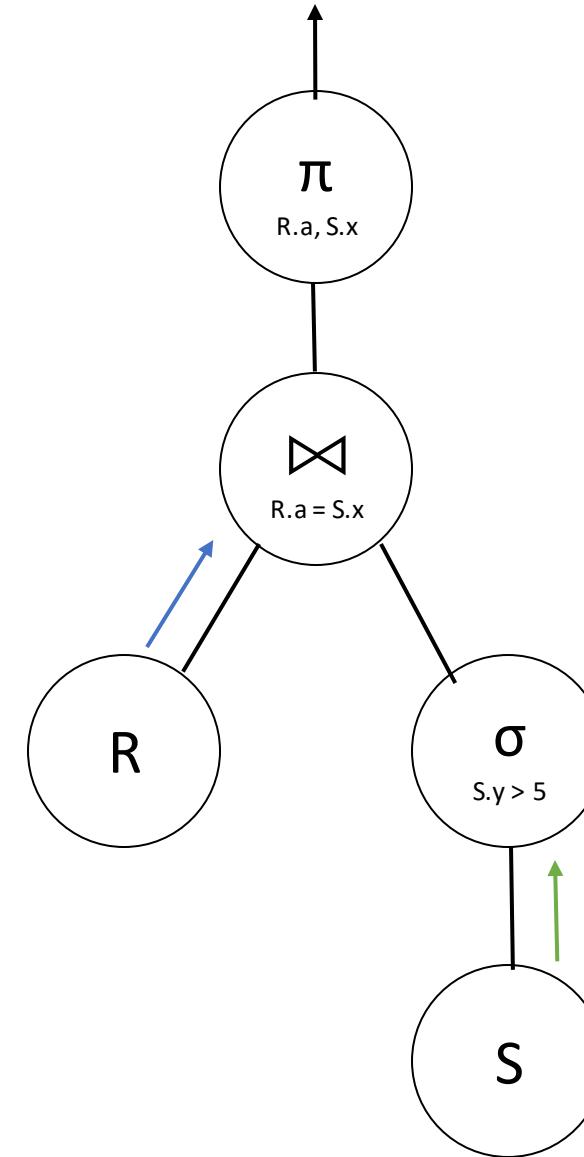
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$

→

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

←

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



Example

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

- Relational Algebra:

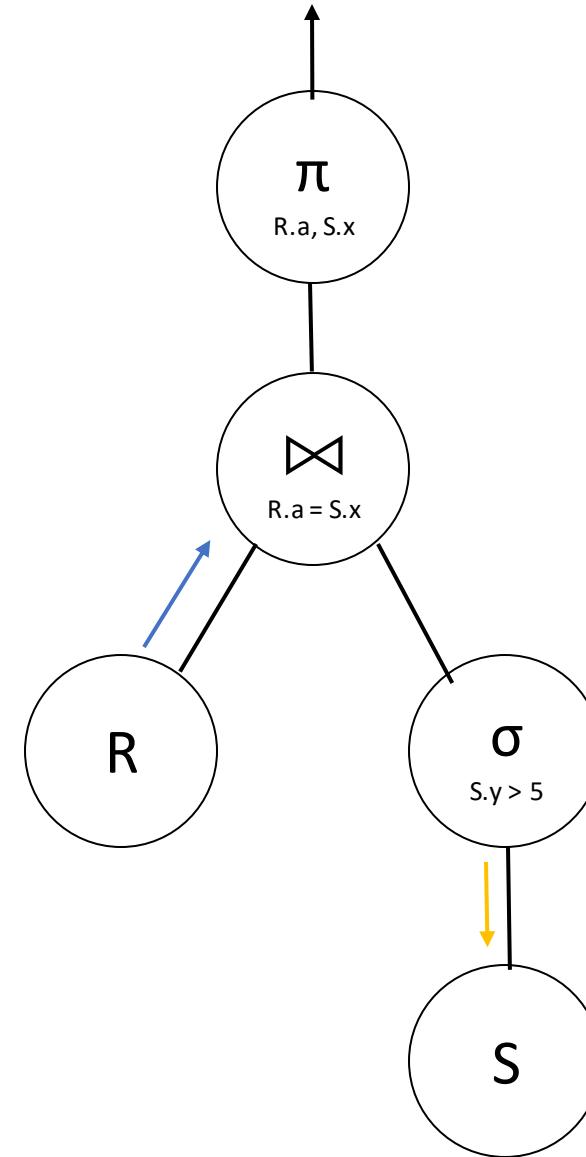
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$



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



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



Example

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

- Relational Algebra:

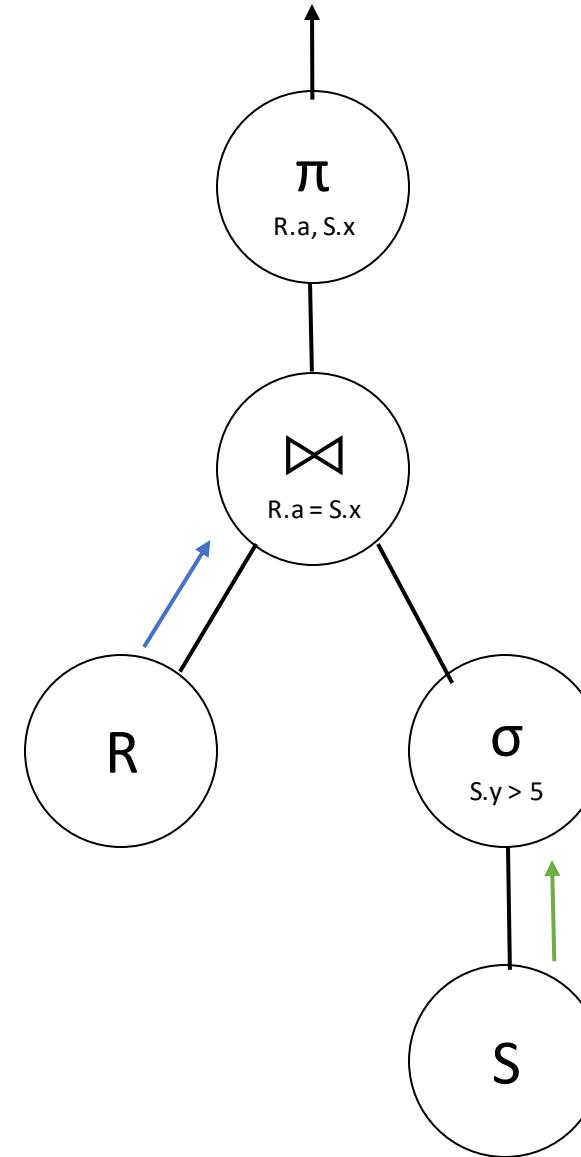
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$

→

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

←

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



Example

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

- Relational Algebra:

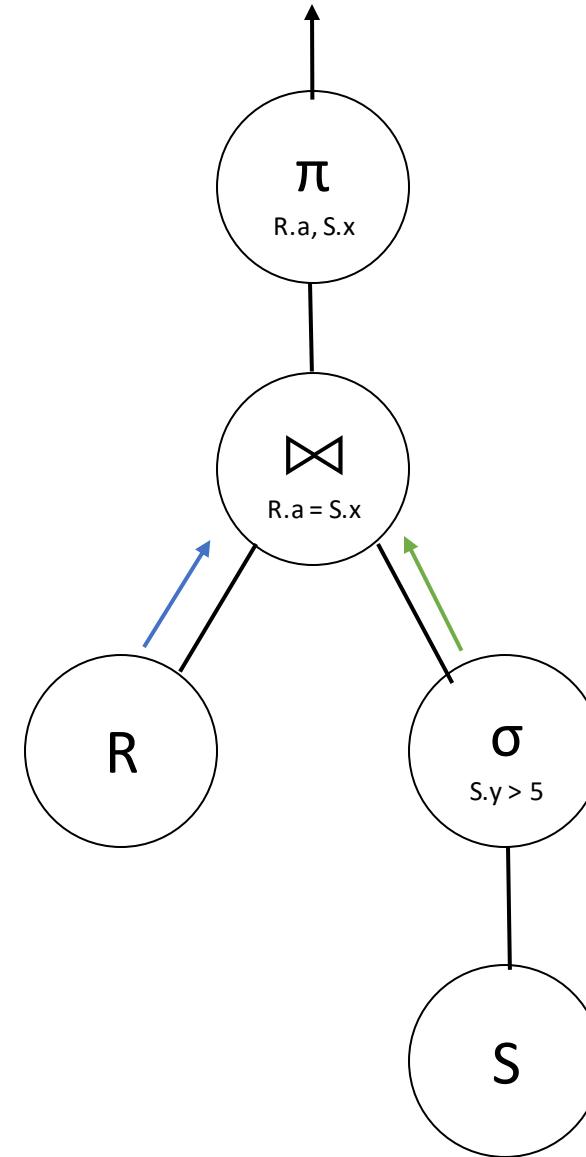
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$

→

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

←

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



Example

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

- Relational Algebra:

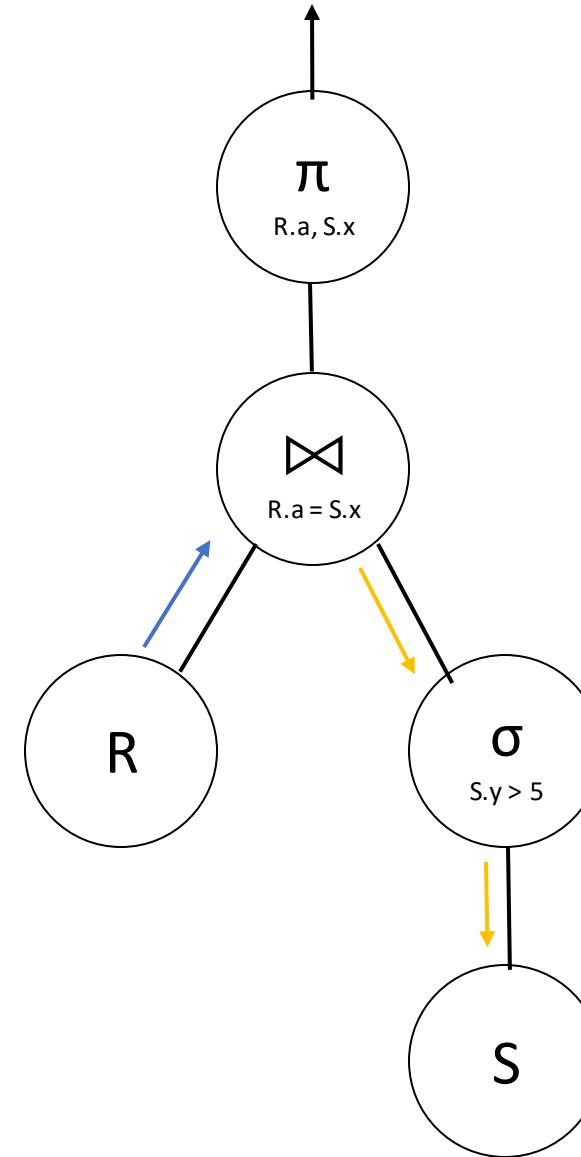
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$



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



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



Example

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

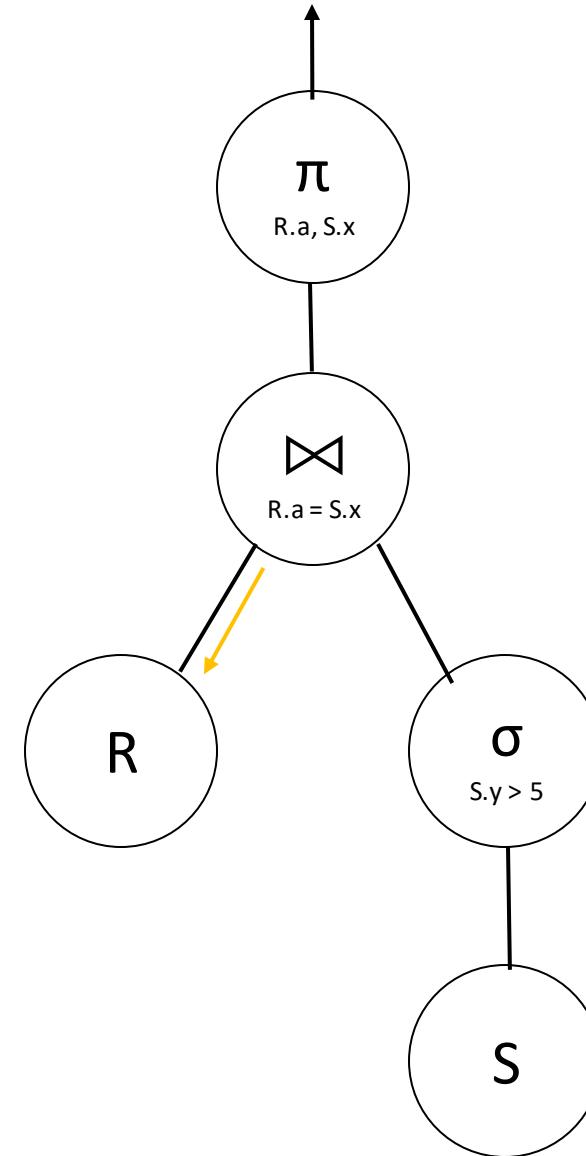
- Relational Algebra:

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



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

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



Example

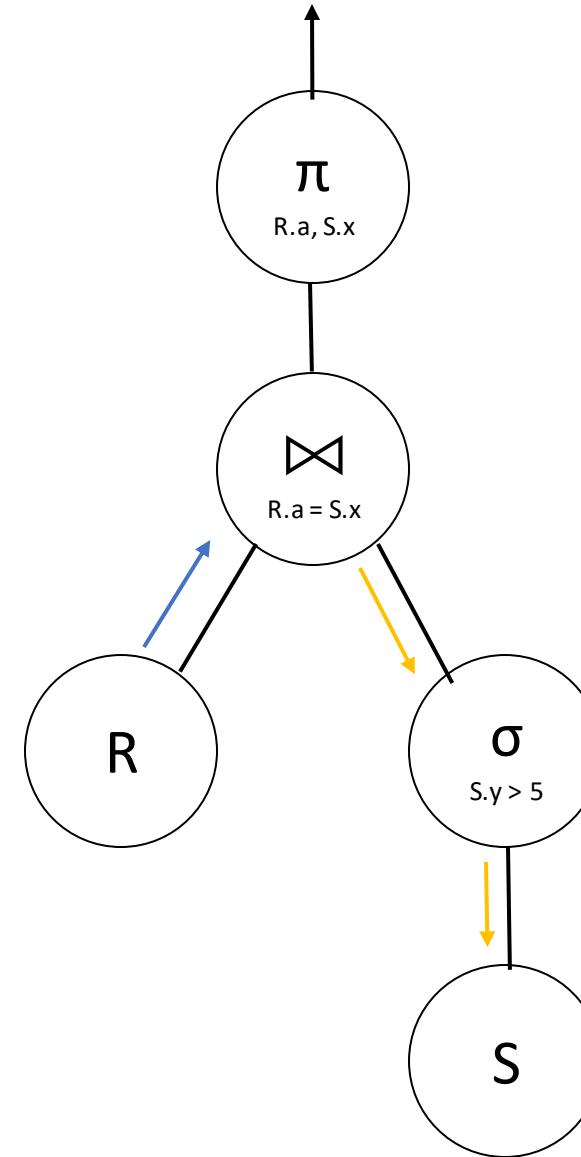
- SELECT R.a, S.x
FROM R, S
WHERE R.a = S.x AND S.y > 5;

- 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



Example

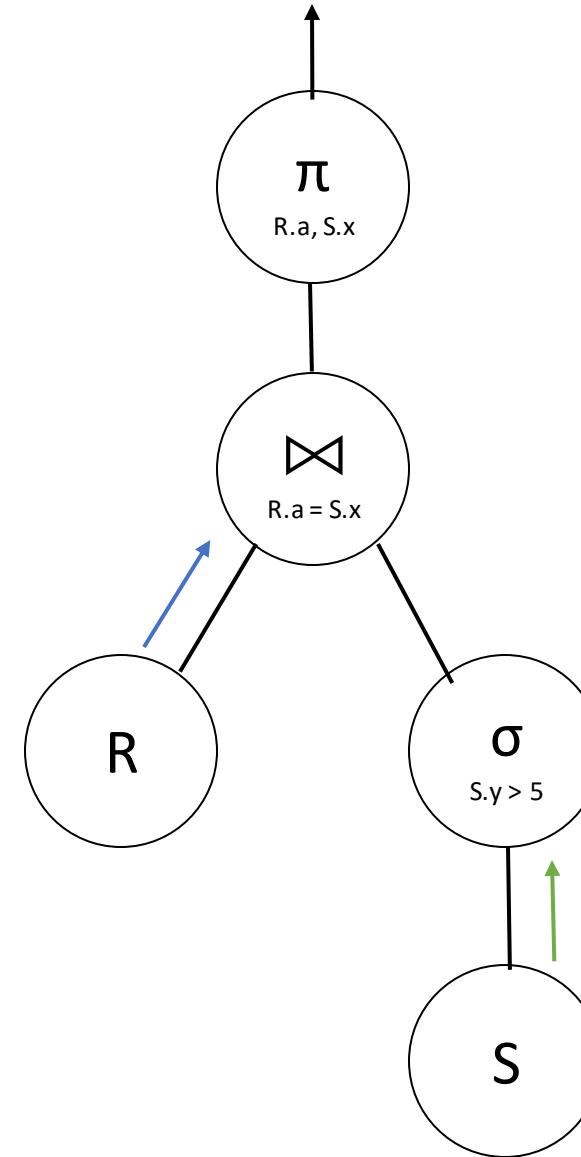
- SELECT R.a, S.x
FROM R, S
WHERE R.a = S.x AND S.y > 5;

- 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



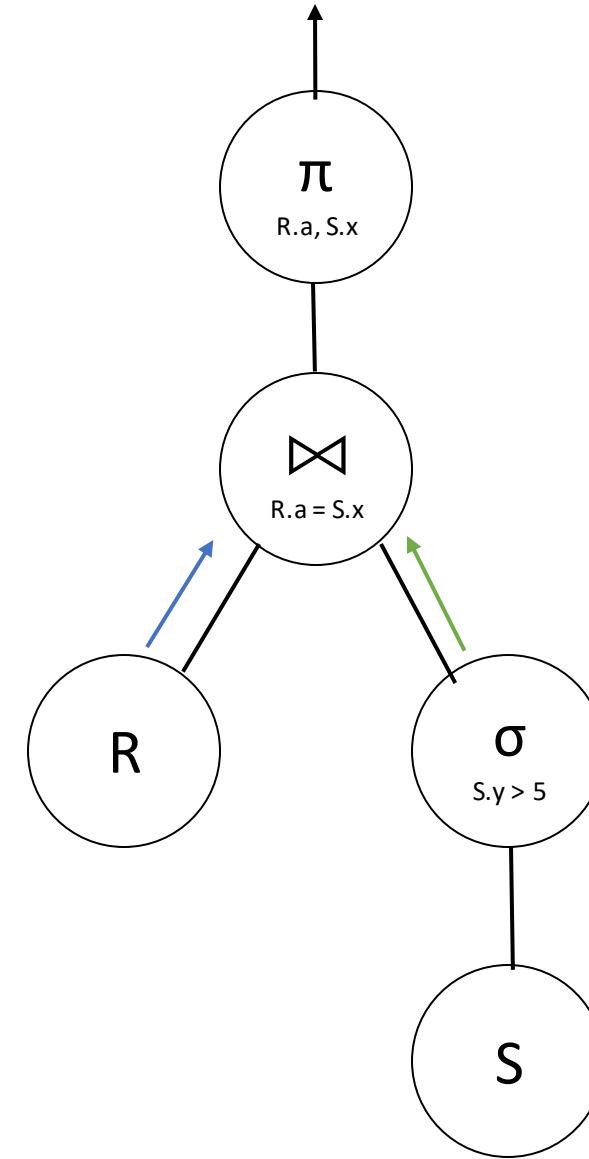
Example

- 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



Example

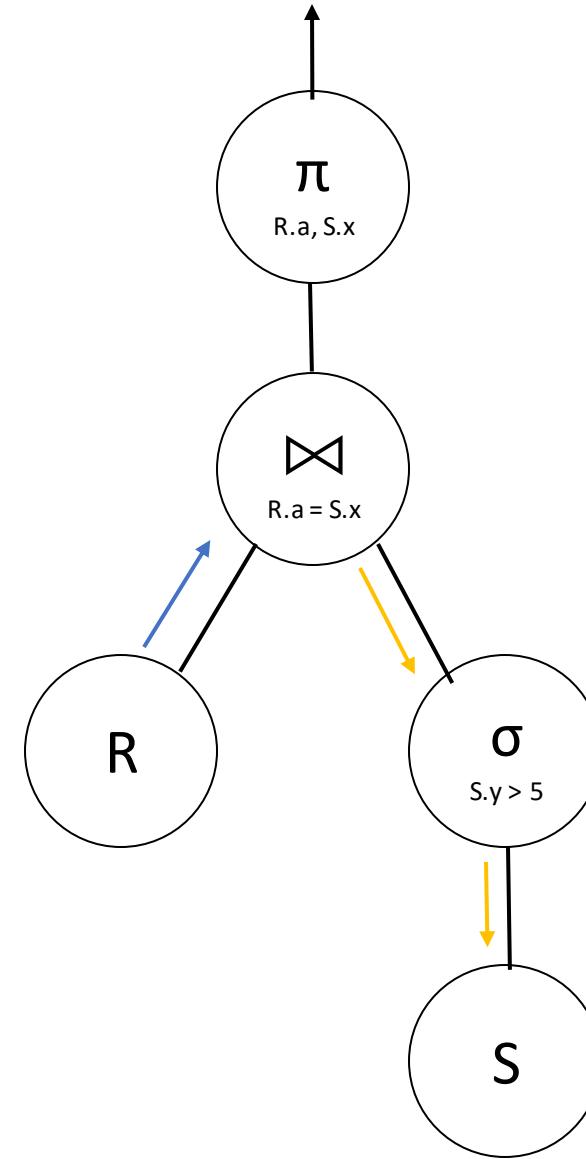
- SELECT R.a, S.x
FROM R, S
WHERE R.a = S.x AND S.y > 5;

- 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



Example

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

- Relational Algebra:

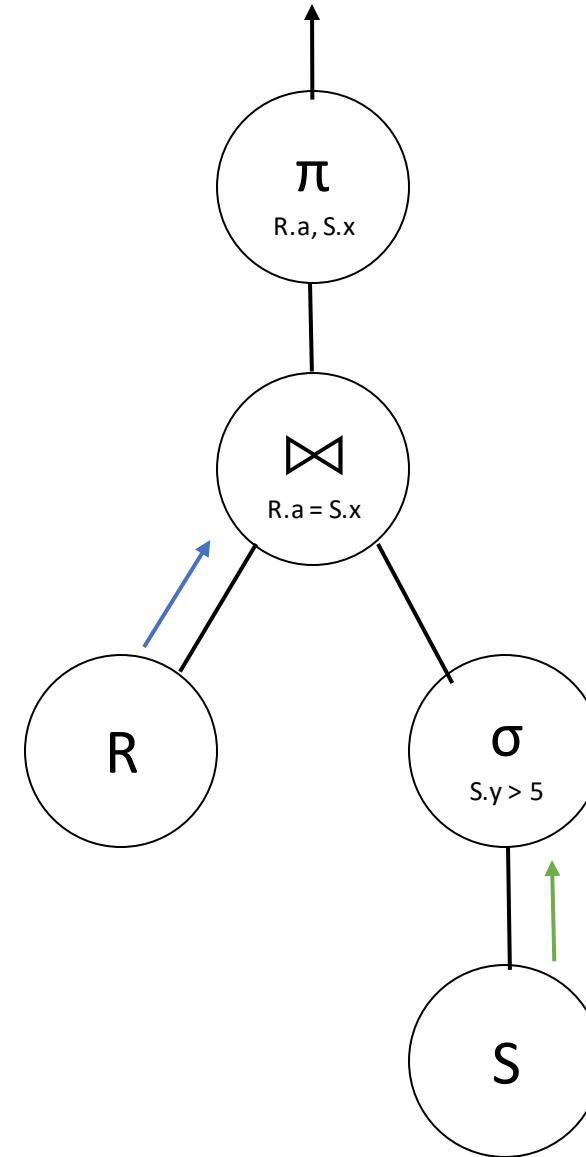
$$\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$$



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



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



Example

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

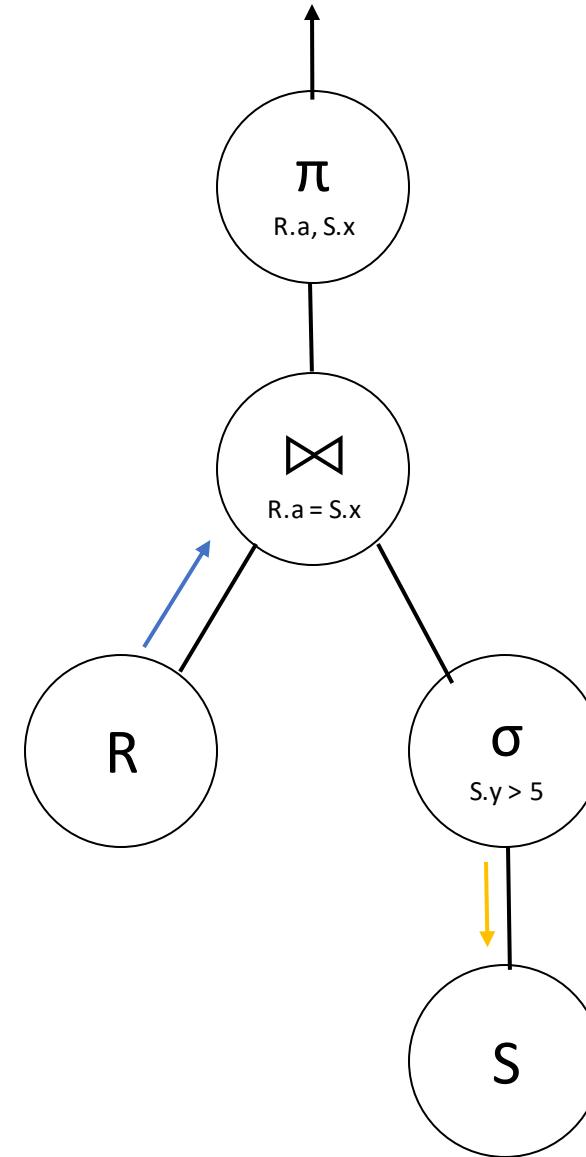
- Relational Algebra:
 $\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$



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



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



Example

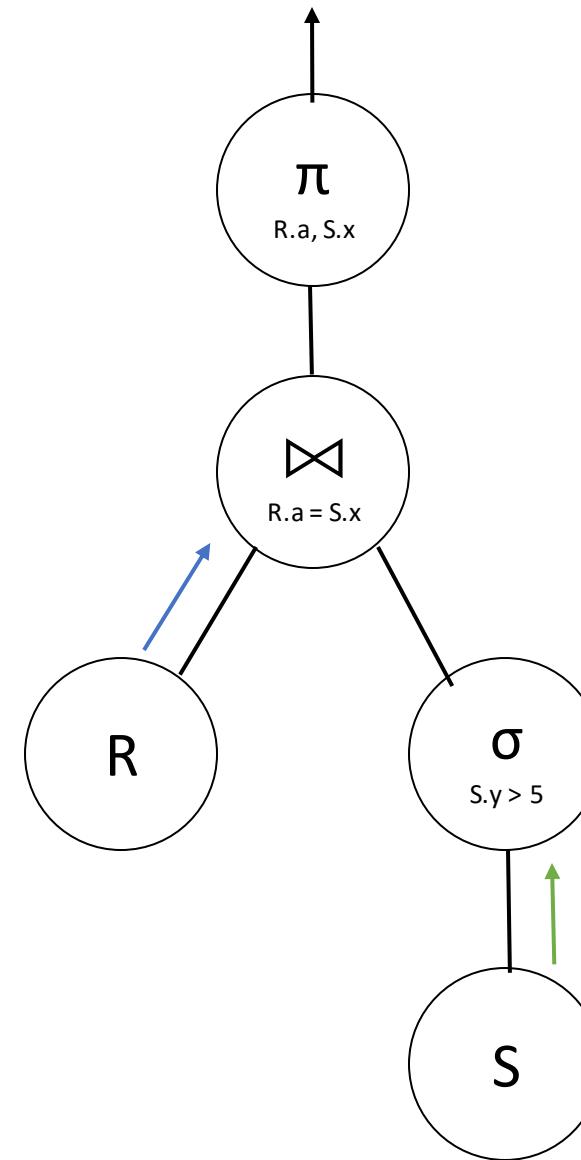
- SELECT R.a, S.x
FROM R, S
WHERE R.a = S.x AND S.y > 5;

- 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



Example

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

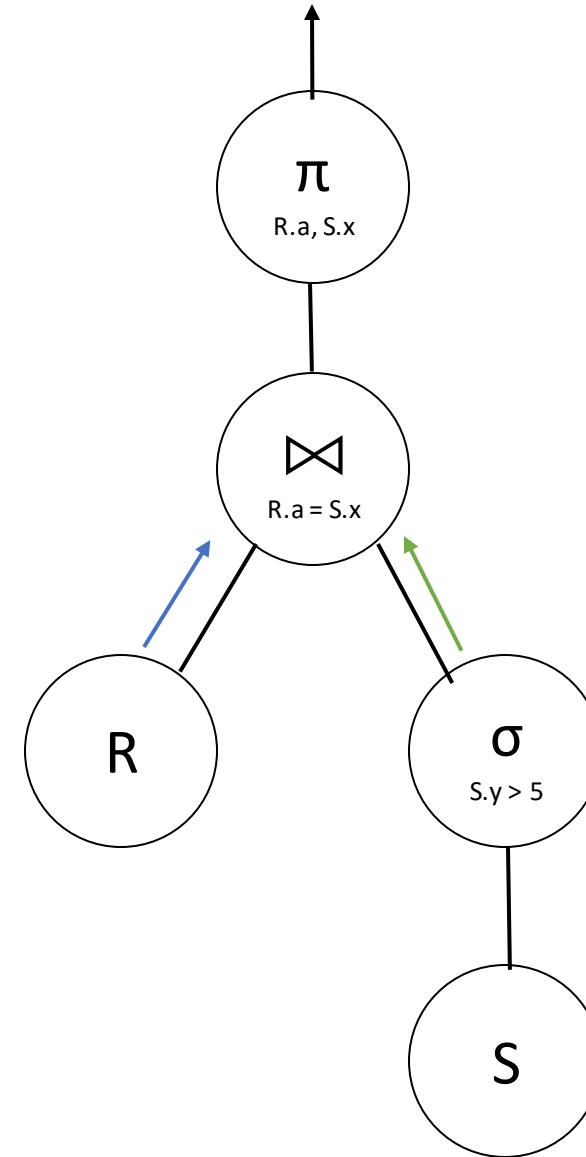
- Relational Algebra:
 $\pi_{R.a, S.x}(\sigma_{S.y > 5}(S) \bowtie_{R.a = S.x} R)$



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



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



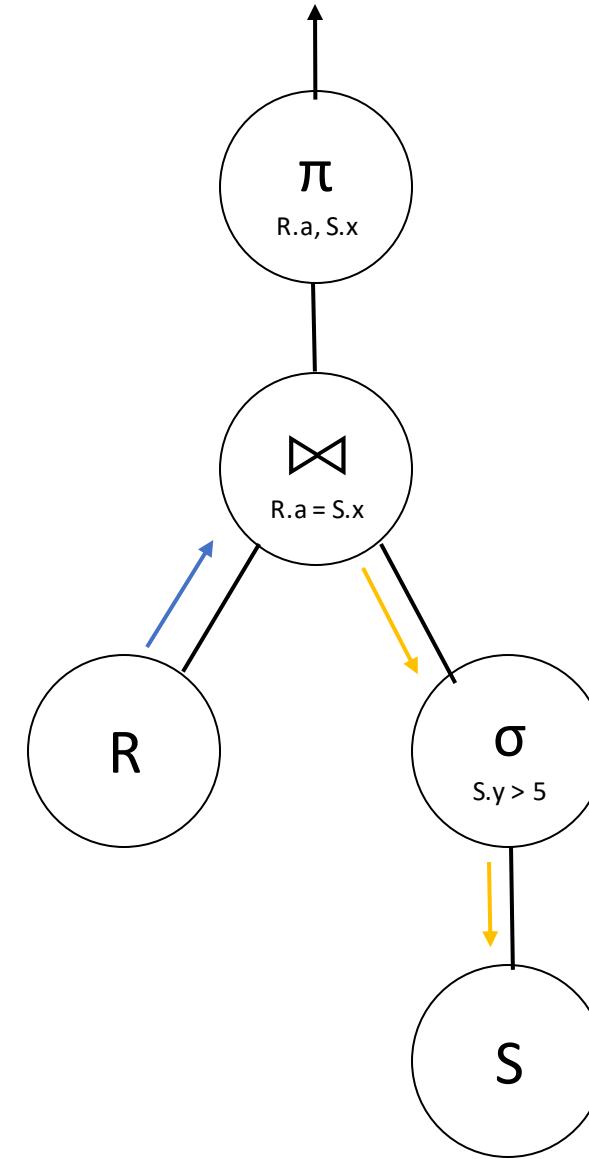
Example

- 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



Example

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

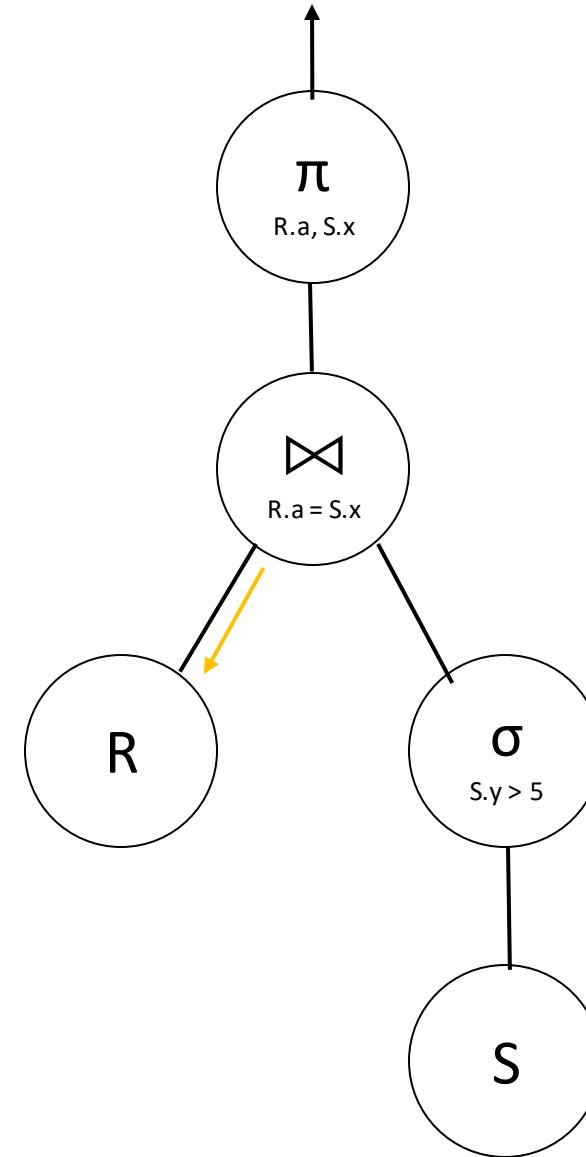
- Relational Algebra:

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



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

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



Example

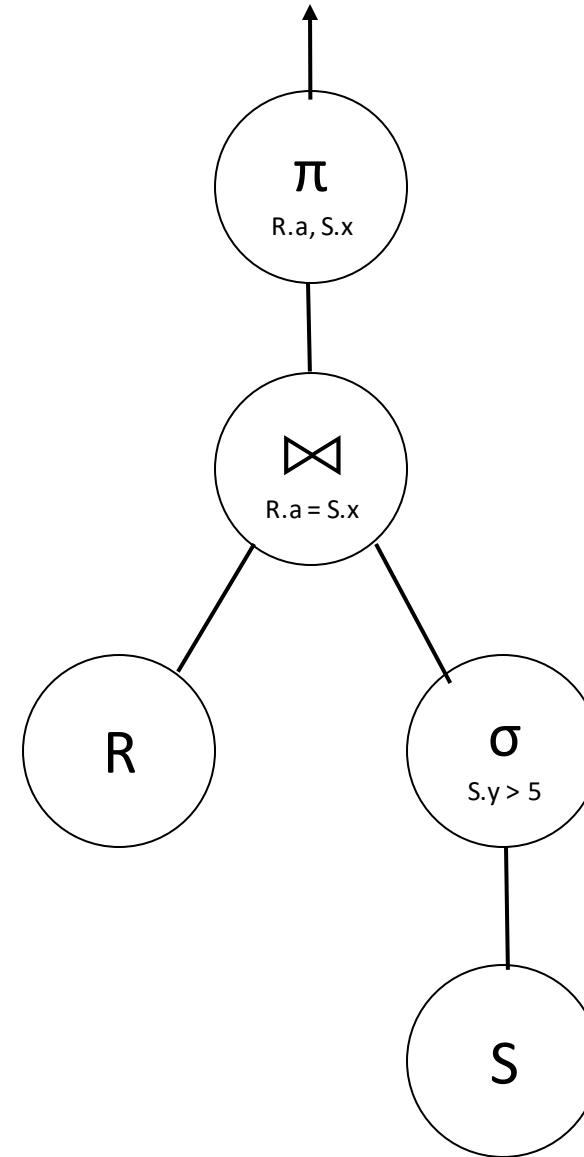
- SELECT R.a, S.x
FROM R, S
WHERE R.a = S.x AND S.y > 5;

- 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



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
- Two phases of a Hash Join:
 1. Construct a Hash Table on the join attribute for the smaller relation (assume it is small enough to fit in memory)
 2. For every tuple in the larger relation, probe the hash table for a match on the join predicate

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
 - **NOTE: 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