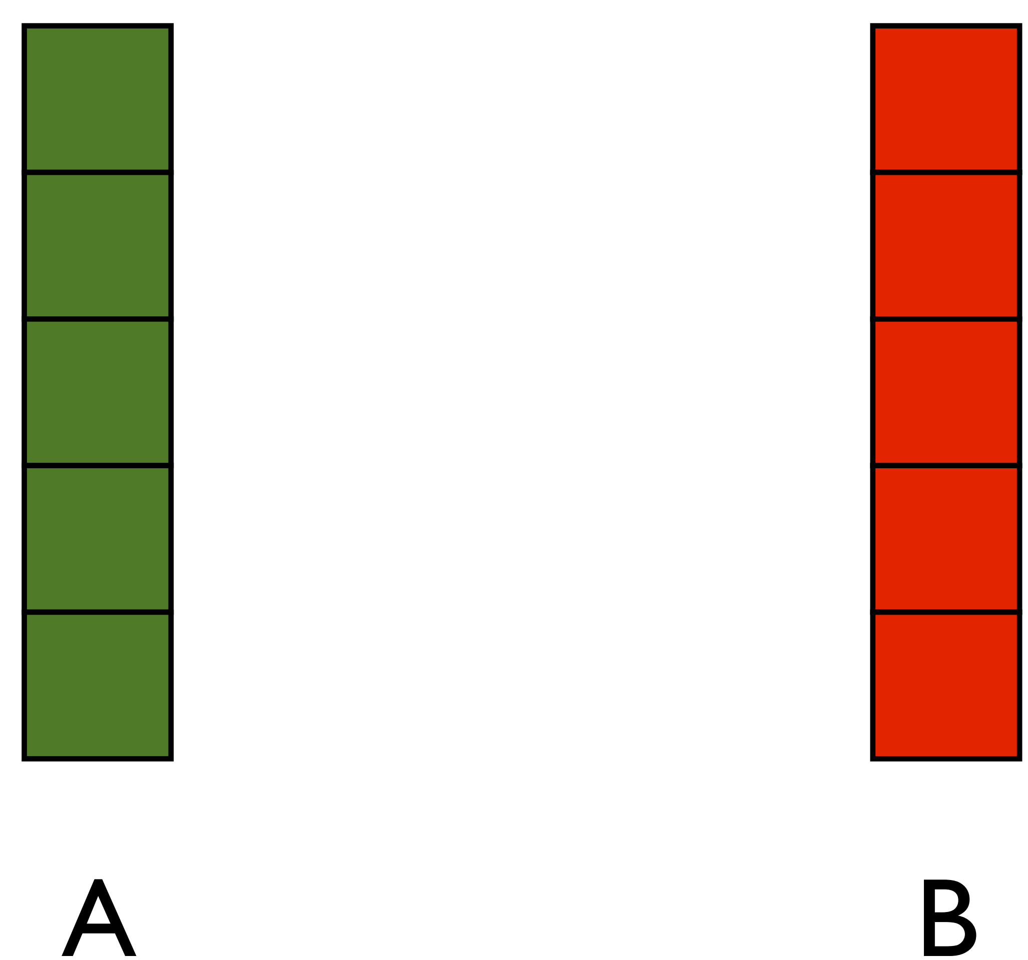
Relational Algebra Equivalencies

Database Systems: The Complete Book
Ch. 16.2-16.3

Implementing: Joins

Solution 1 (Nested-Loop)

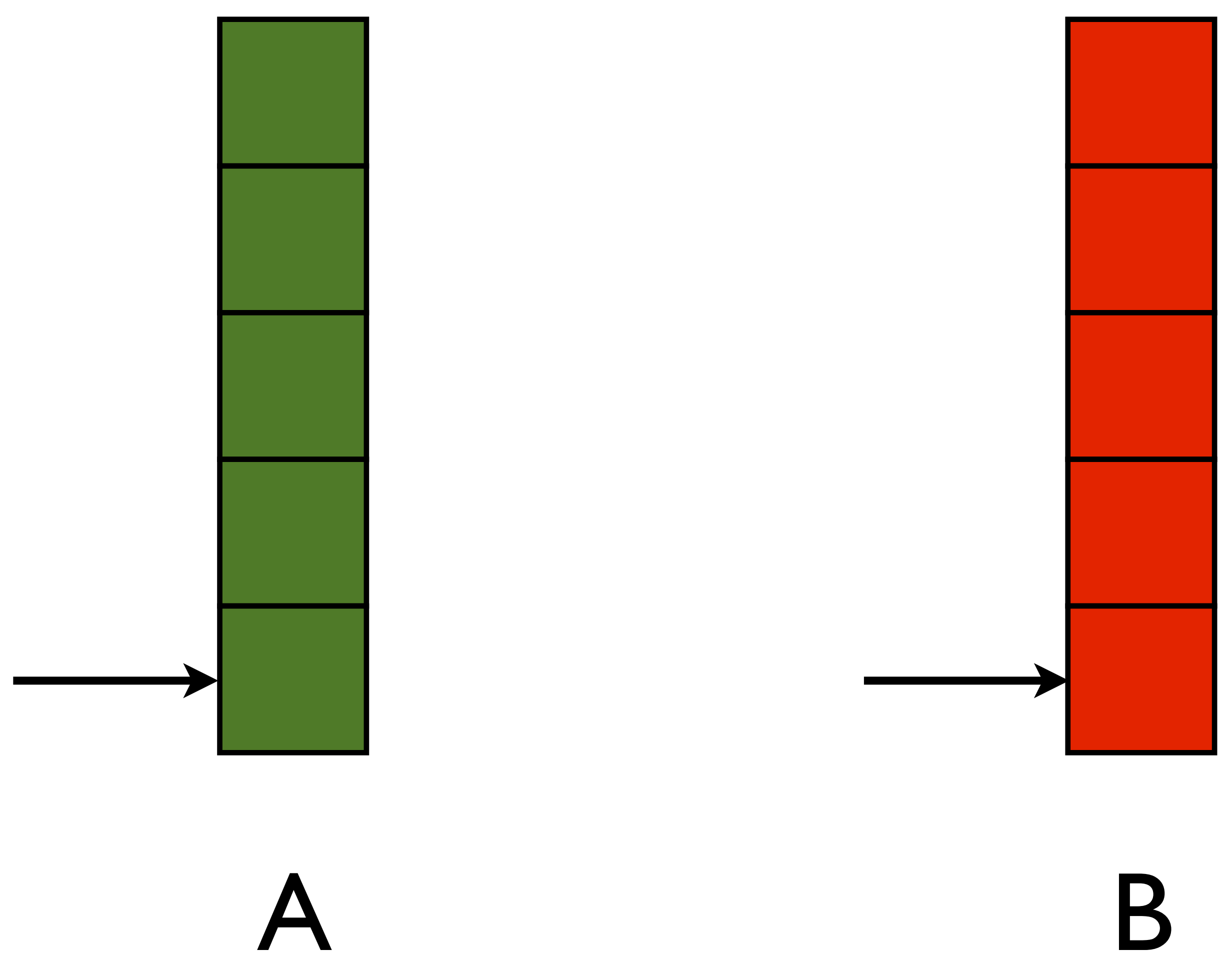
For Each (a in A) { For Each (b in B) { emit (a, b); }}



Implementing: Joins

Solution 1 (Nested-Loop)

For Each (a in A) { For Each (b in B) { emit (a, b); }}



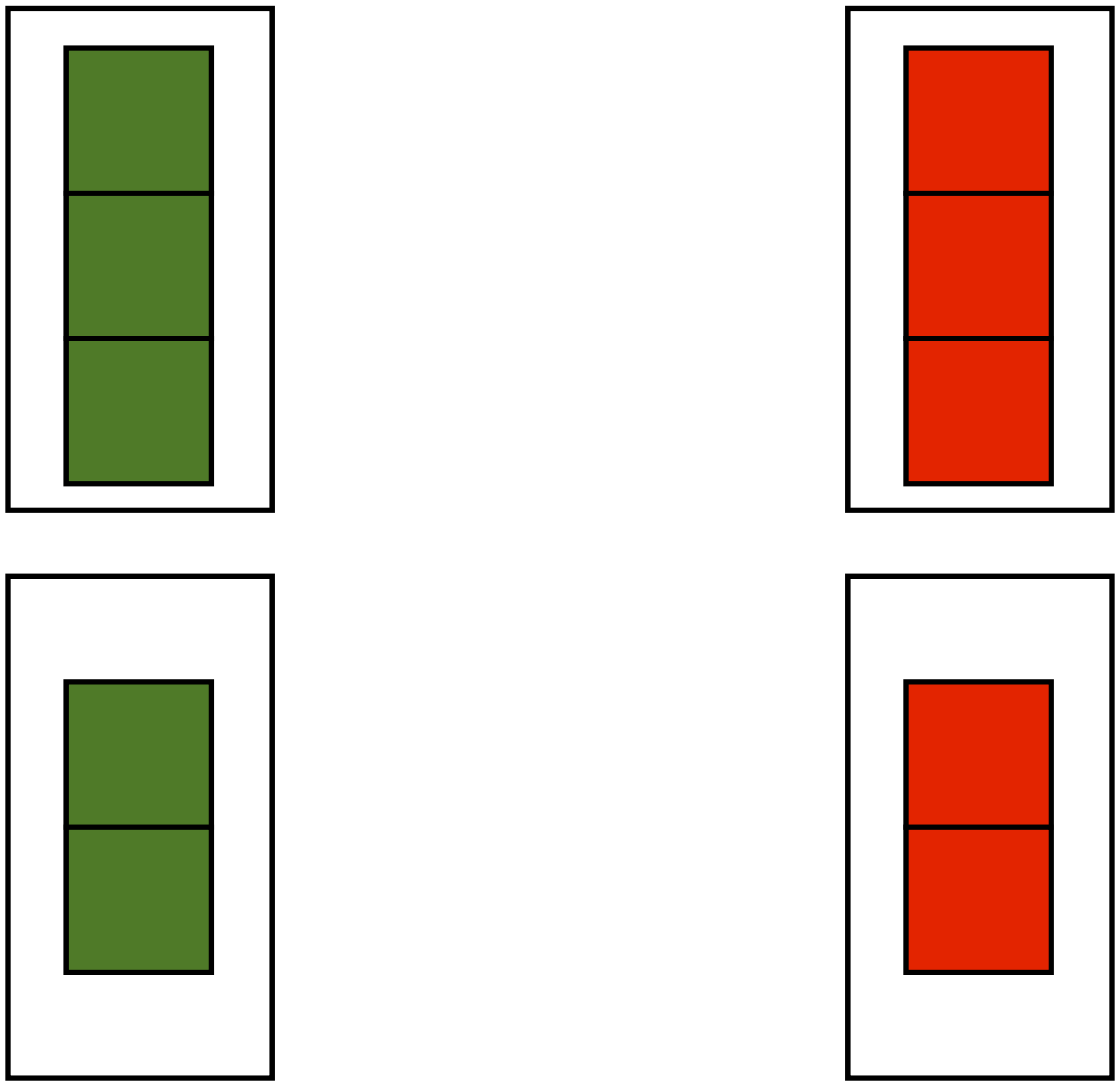
Implementing: Joins

Solution 2 (Block-Nested-Loop)

Implementing: Joins

Solution 2 (Block-Nested-Loop)

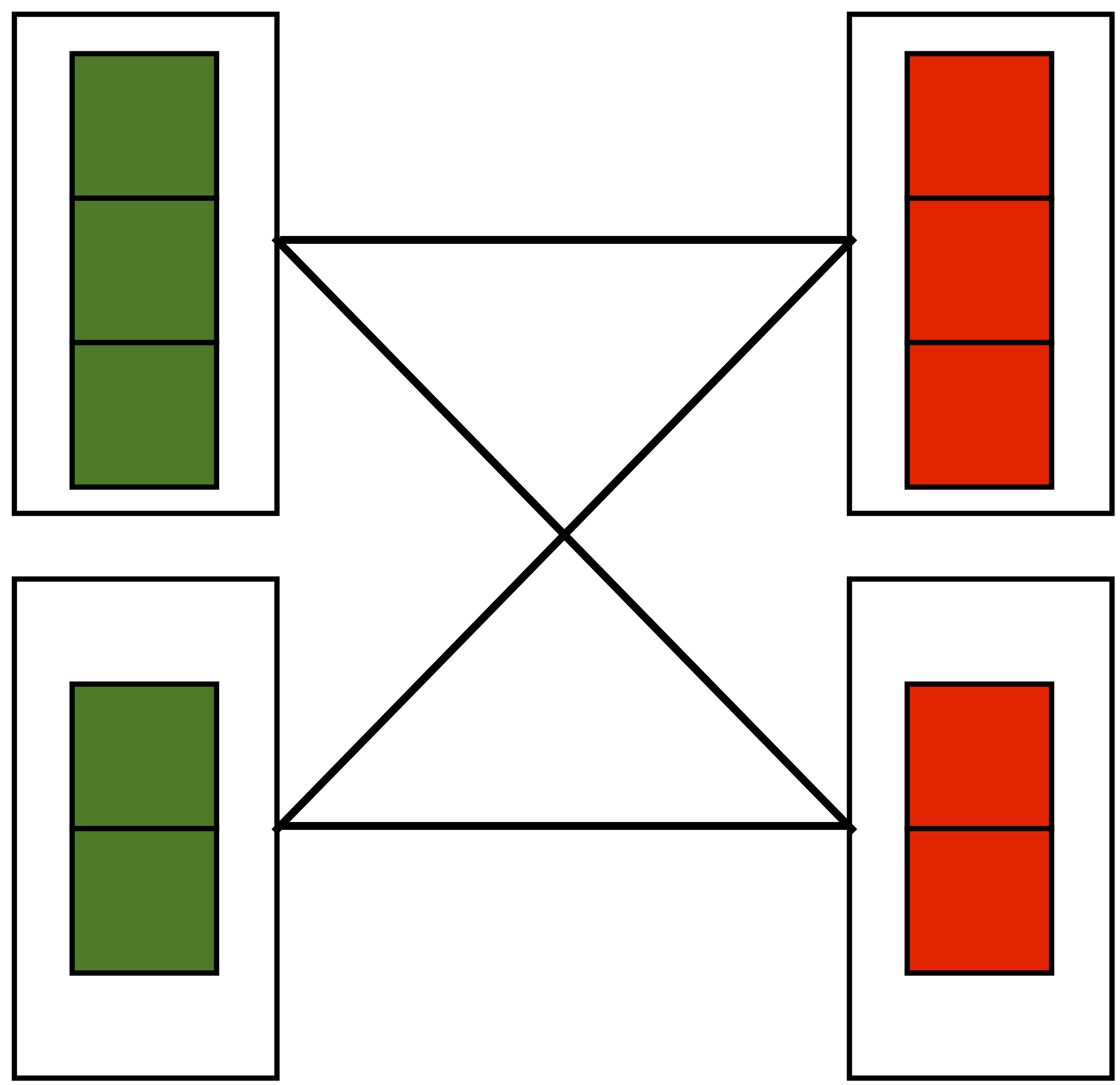
1) Partition into Blocks



Implementing: Joins

Solution 2 (Block-Nested-Loop)

1) Partition into Blocks 2) NLJ on each pair of blocks

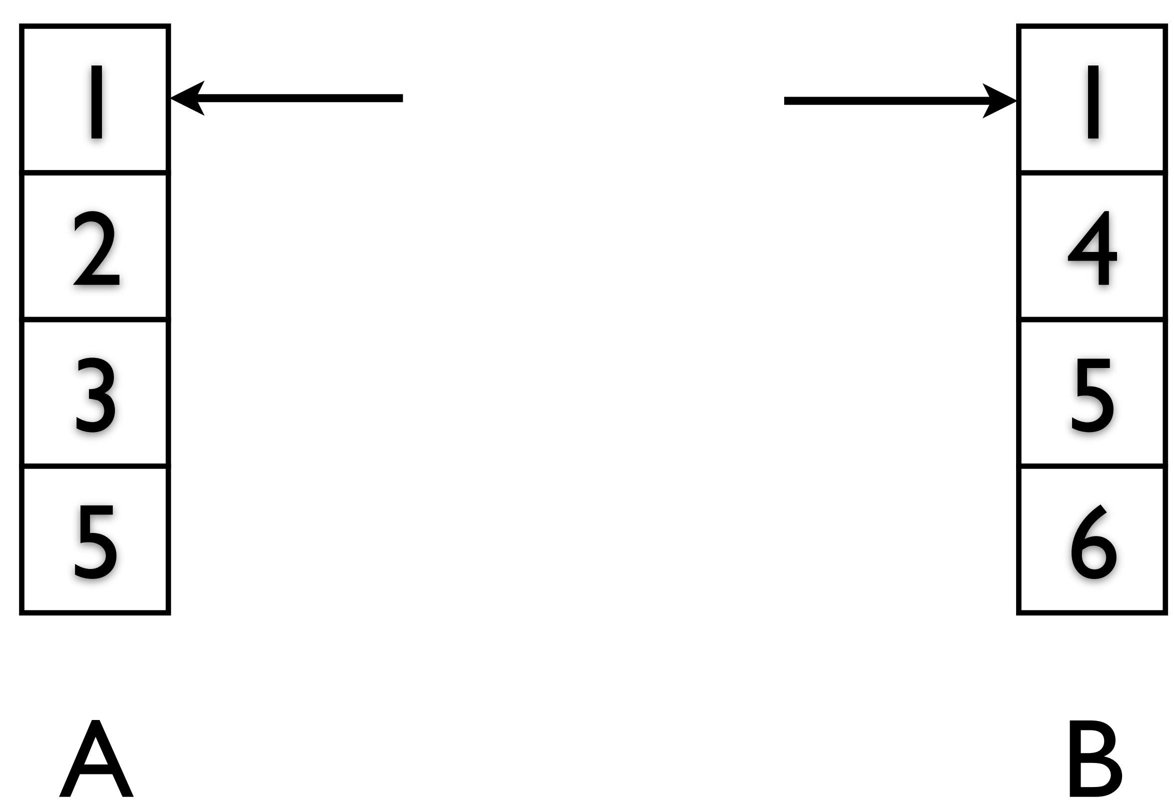


Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value.

When you hit two that match, emit, then iterate both

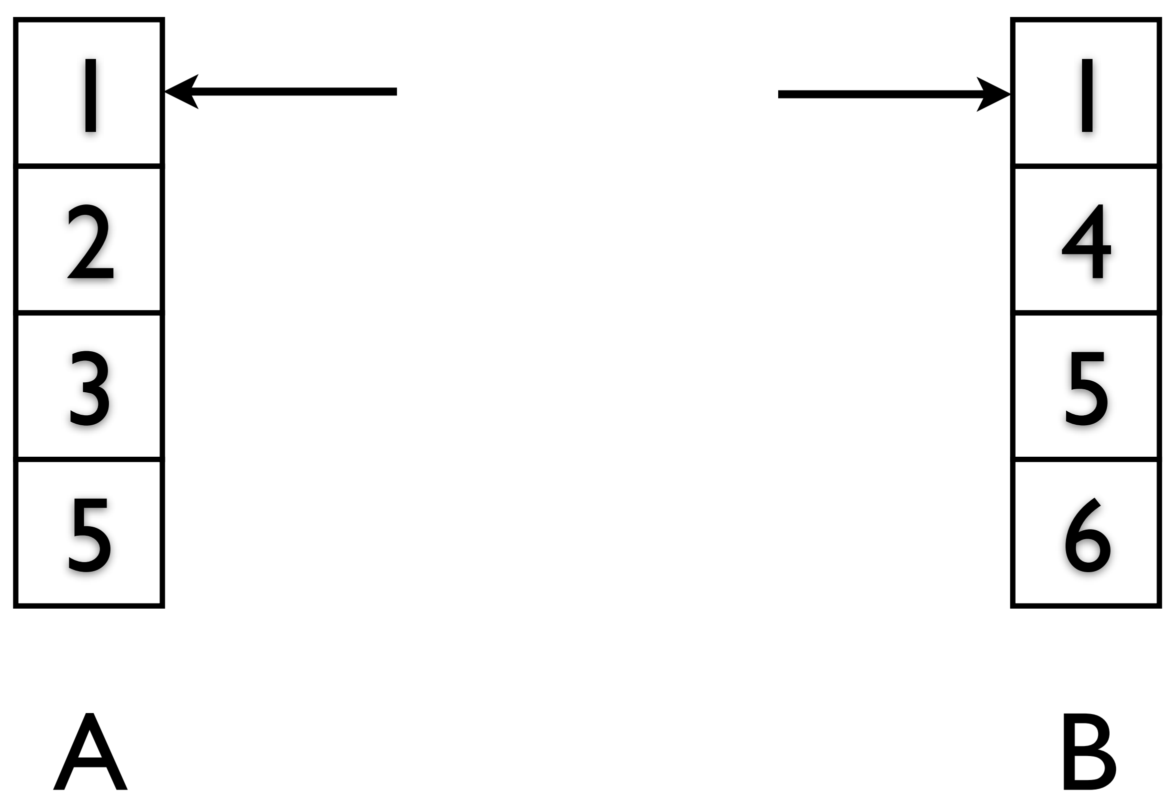


Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value.

When you hit two that match, emit, then iterate both

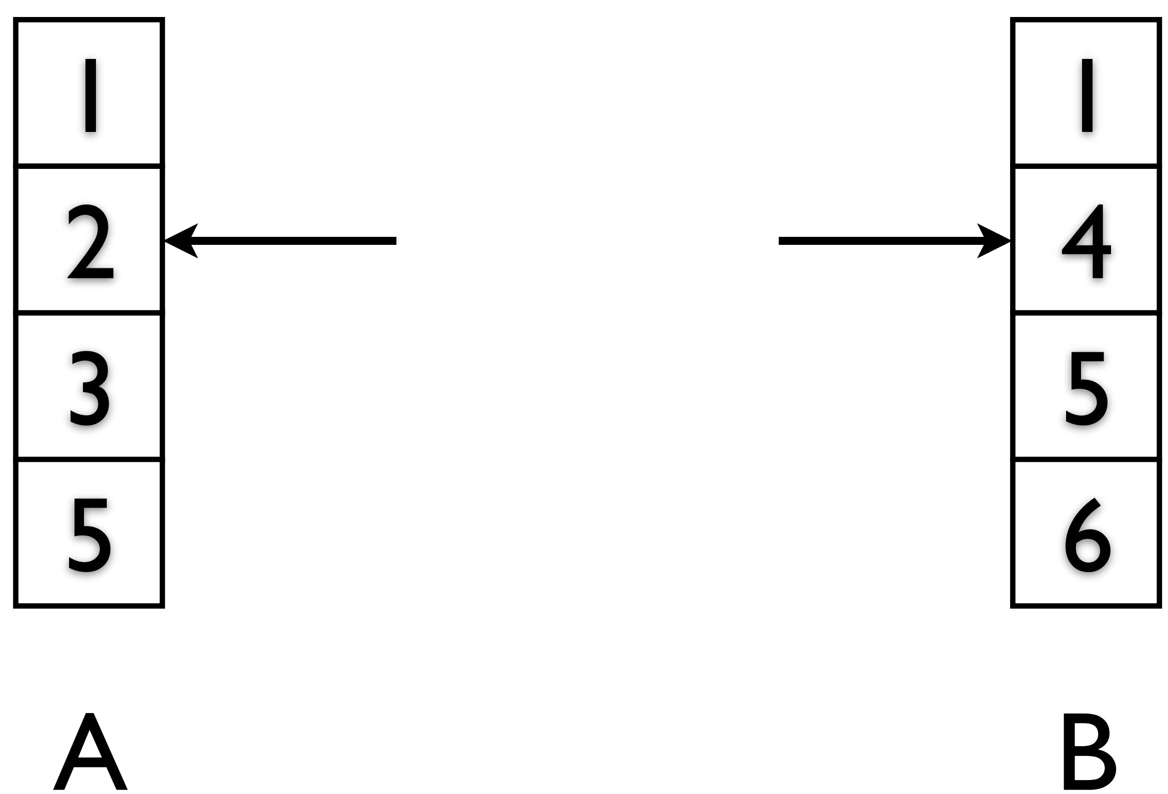


Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value.

When you hit two that match, emit, then iterate both

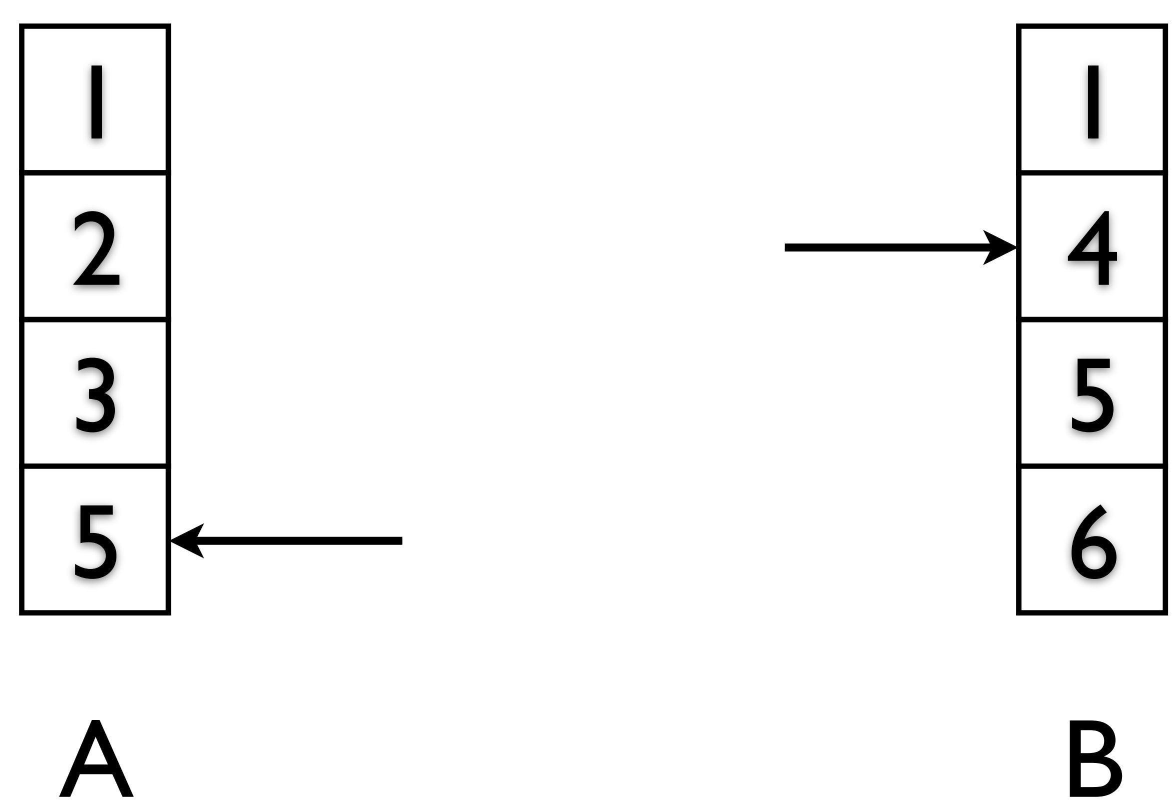


Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value.

When you hit two that match, emit, then iterate both

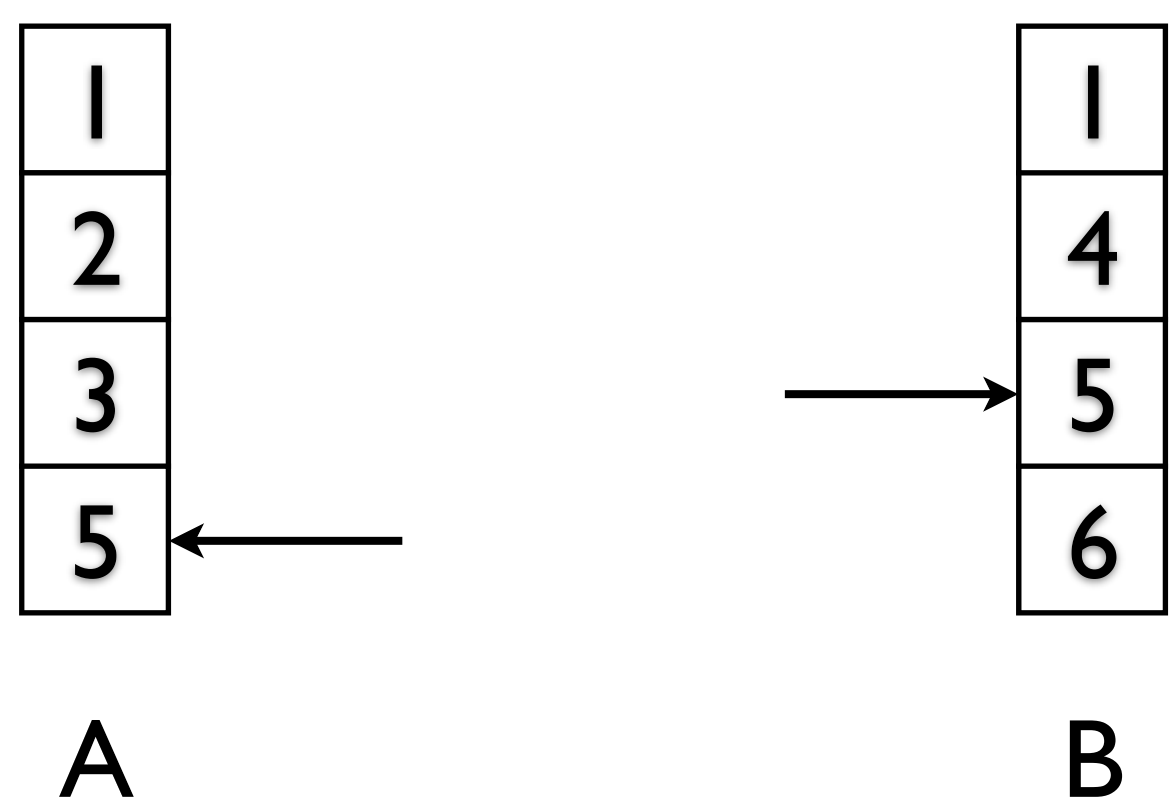


Implementing: Joins

Solution 3 (Sort-Merge Join)

Keep iterating on the set with the lowest value.

When you hit two that match, emit, then iterate both

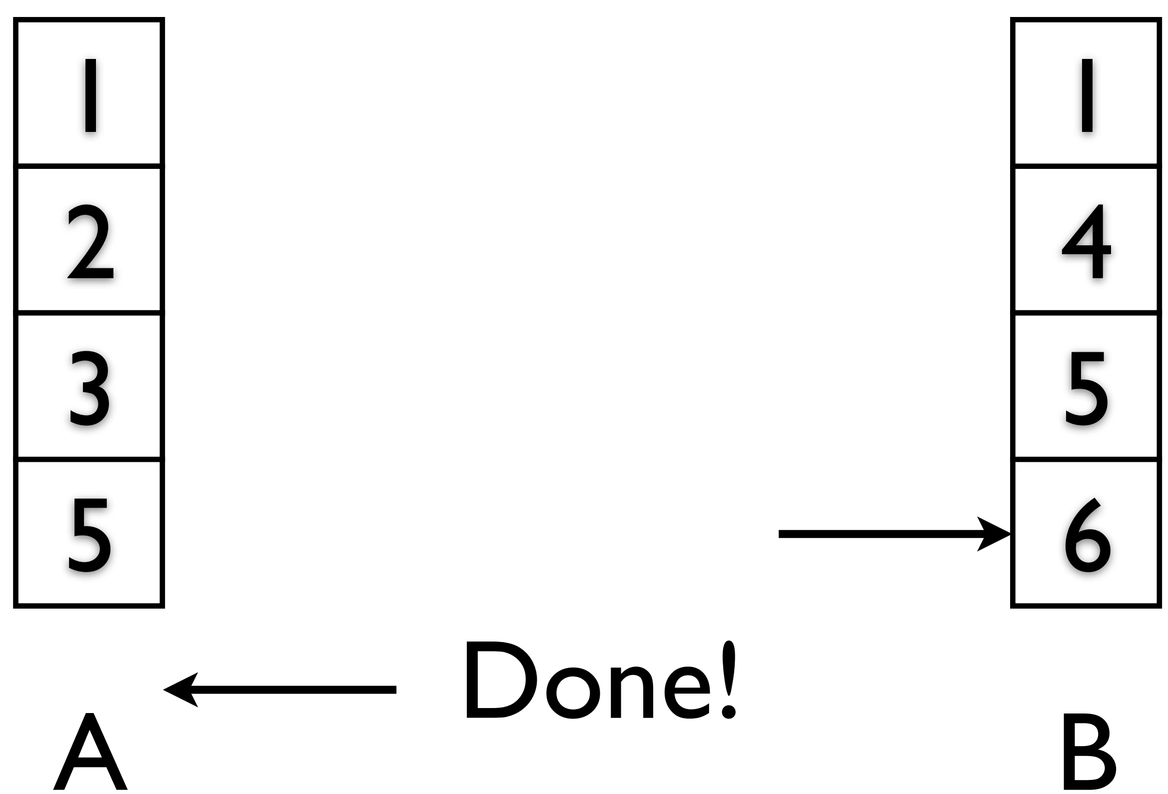


Implementing: Joins

Solution 3 (Sort-Merge Join)

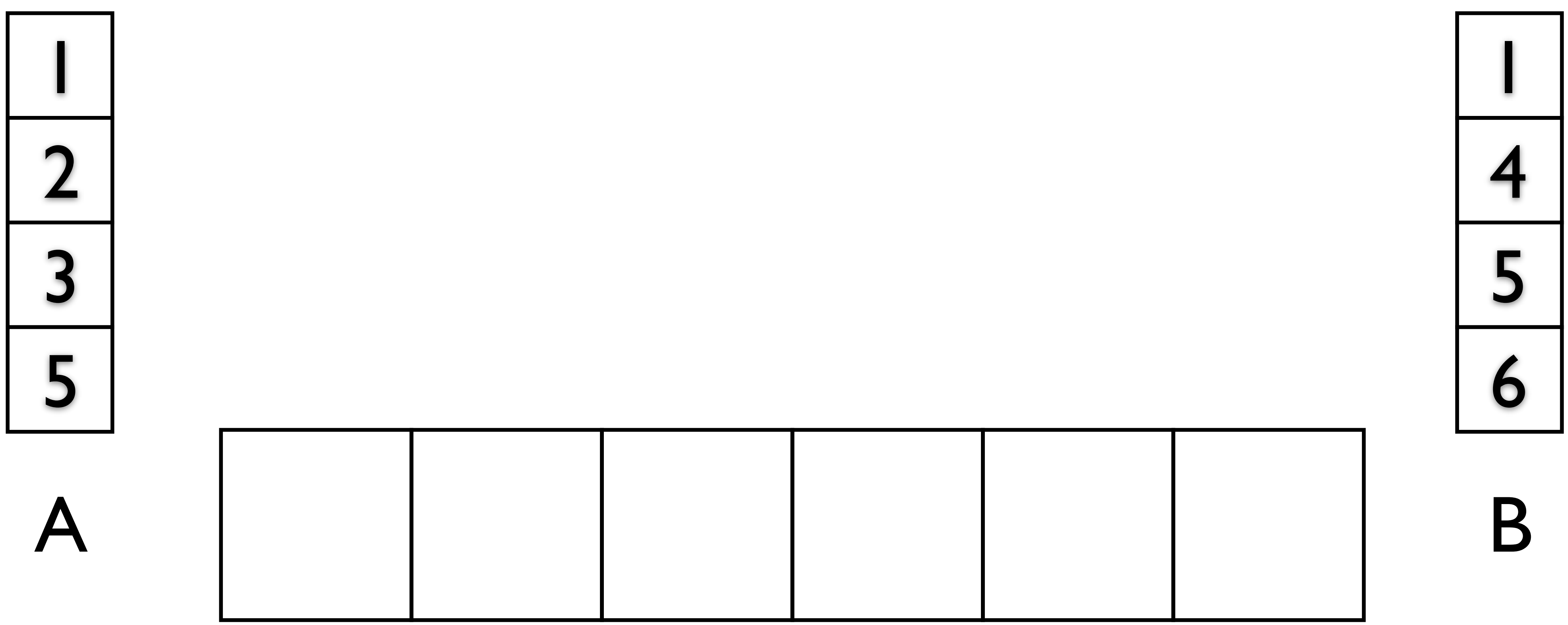
Keep iterating on the set with the lowest value.

When you hit two that match, emit, then iterate both



Implementing: Joins

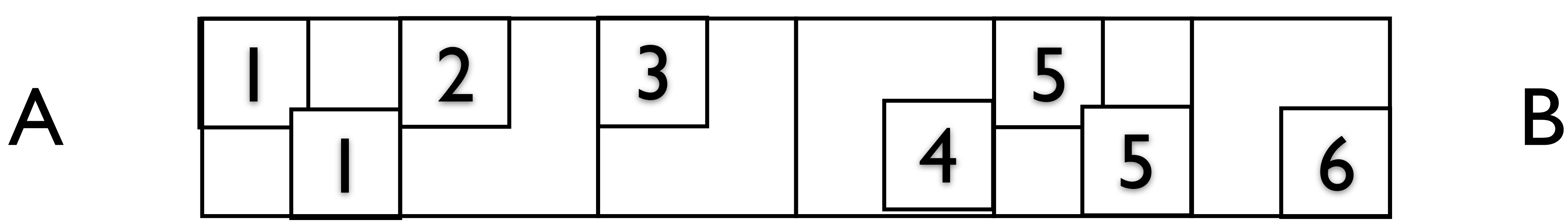
Solution 4 (External Hash)



Implementing: Joins

Solution 4 (External Hash)

1) Build a hash table on both relations



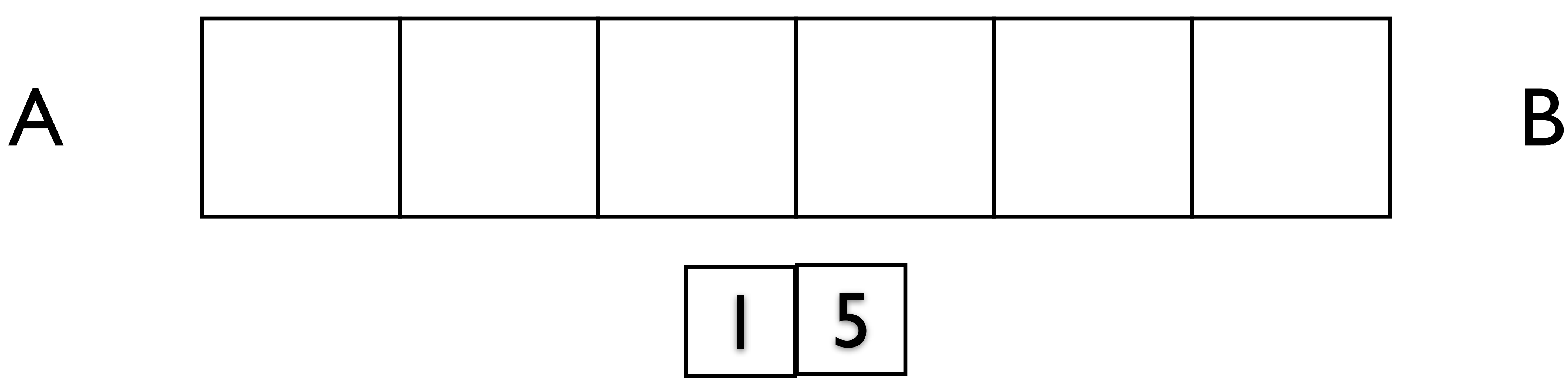
Implementing: Joins

Solution 4 (External Hash)

1) Build a hash table on both relations

2) In-Memory Nested-Loop Join on each hash bucket

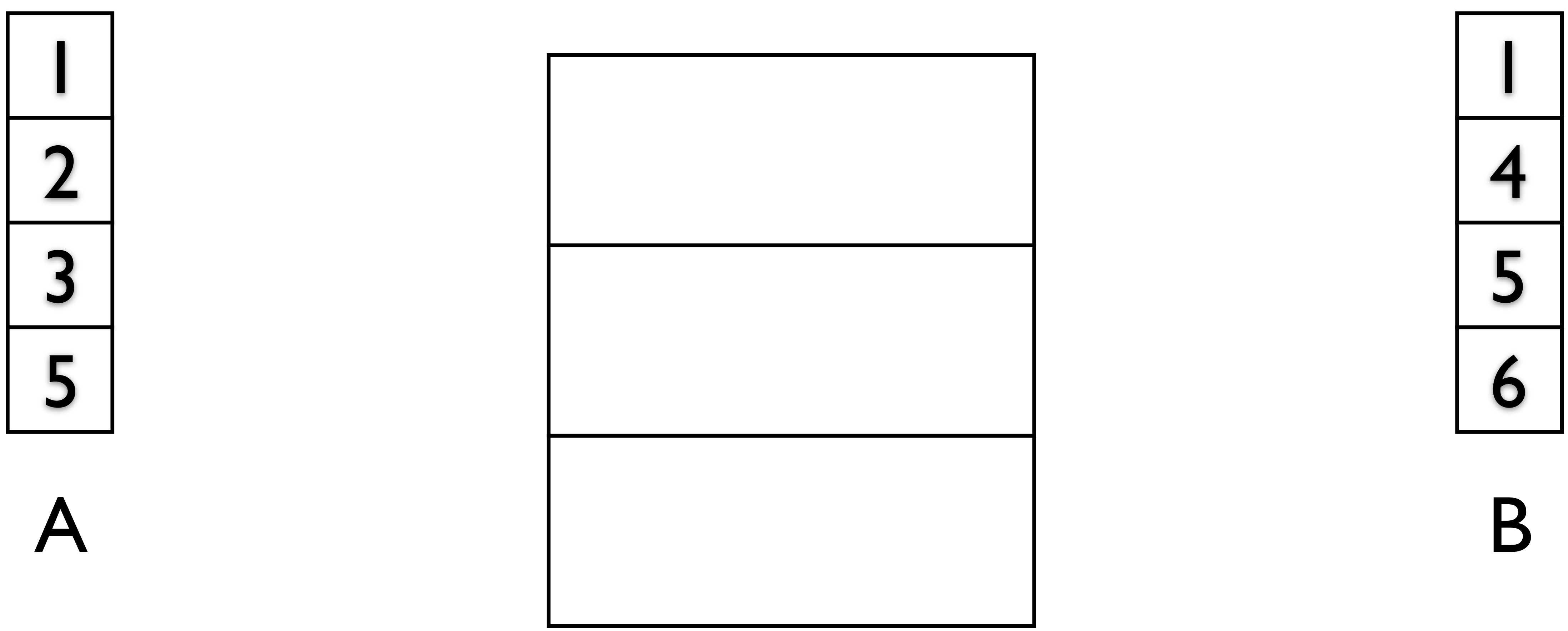
(subdivide buckets using a different hash fn if needed)



Implementing: Joins

Solution 5 (Grace/Hybrid Hash)

Keep the hash table in memory

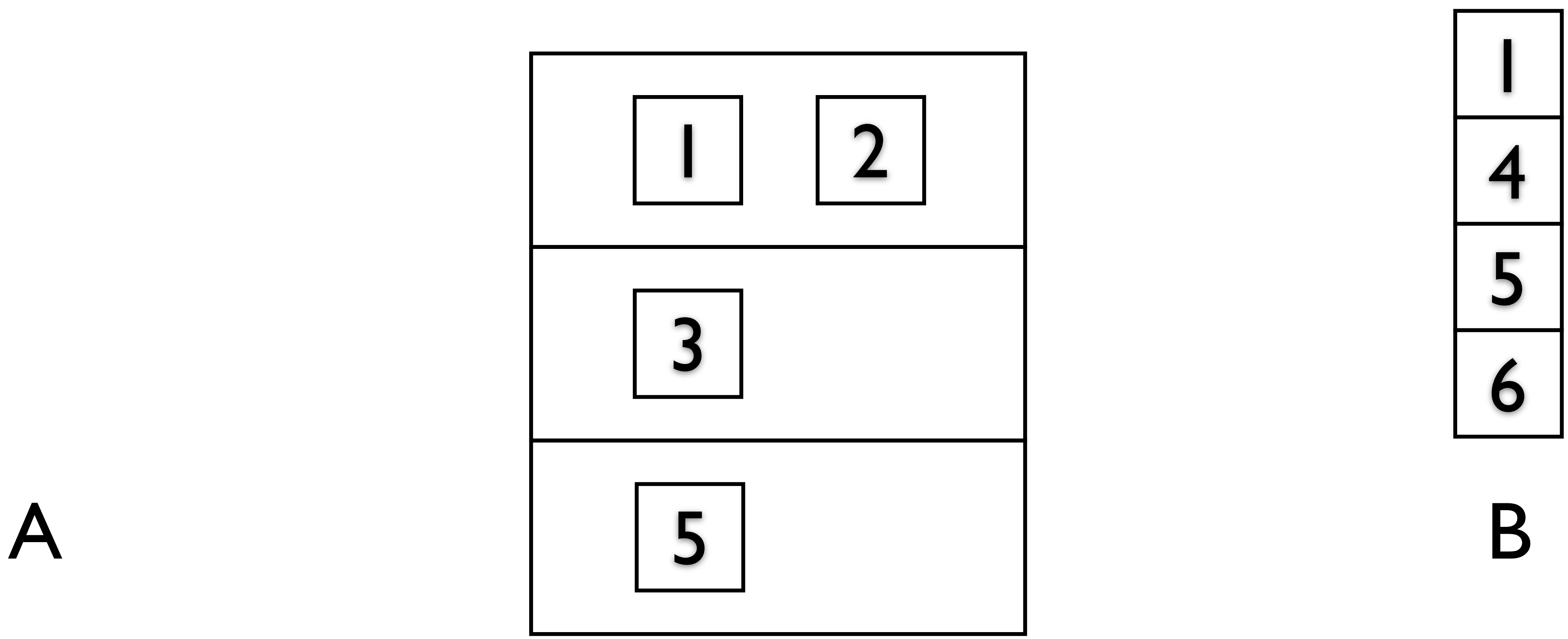


(Essentially a more efficient nested loop join)

Implementing: Joins

Solution 5 (Grace/Hybrid Hash)

Keep the hash table in memory

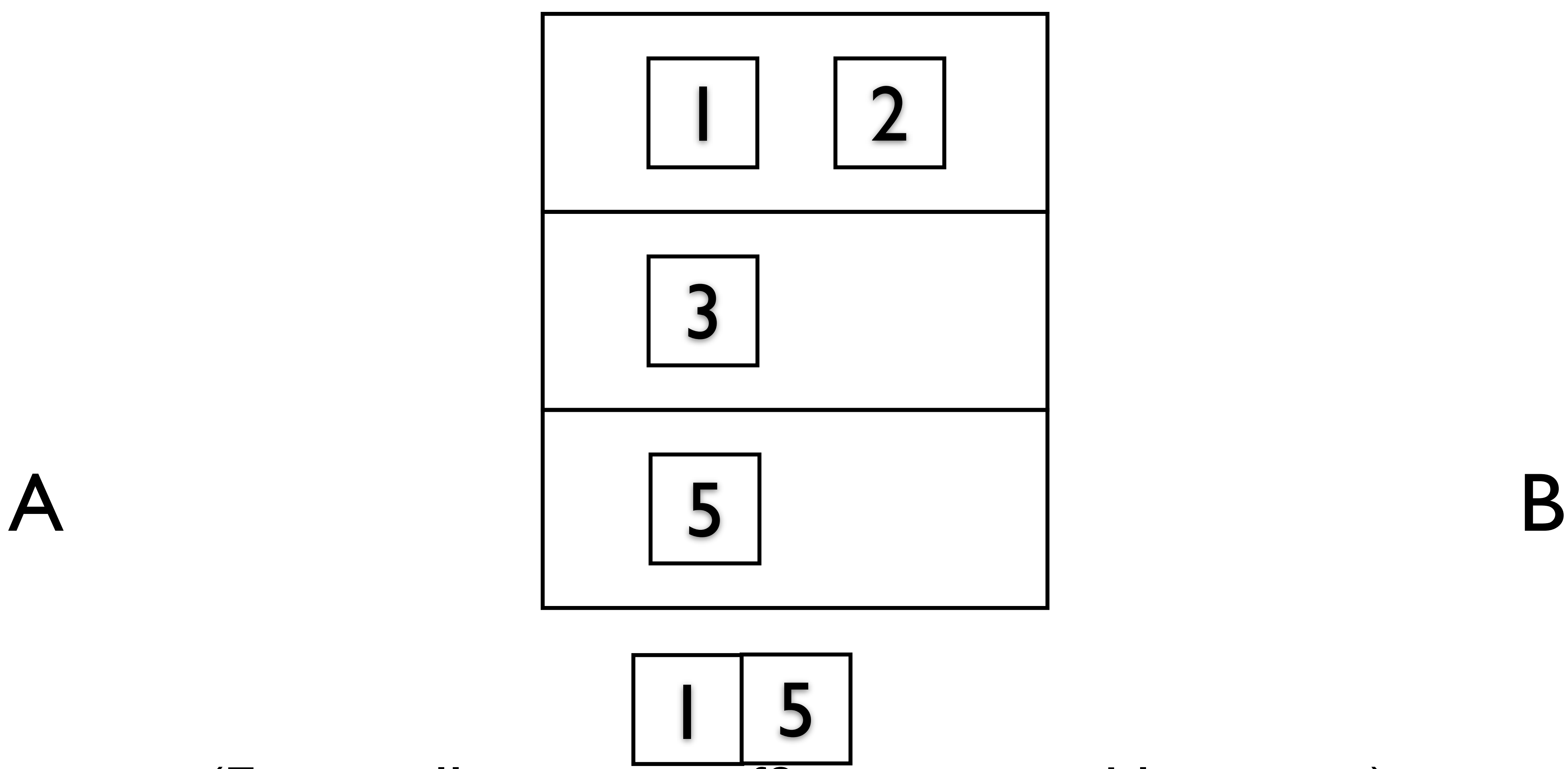


(Essentially a more efficient nested loop join)

Implementing: Joins

Solution 5 (Grace/Hybrid Hash)

Keep the hash table in memory

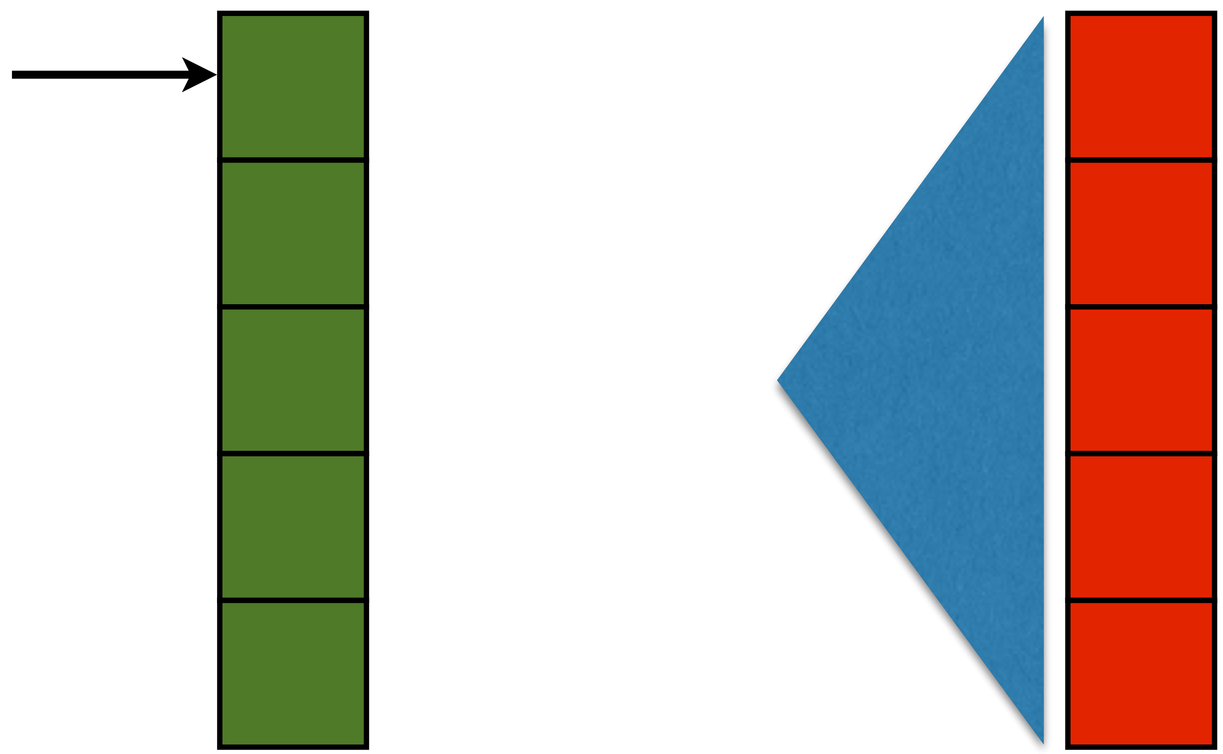


(Essentially a more efficient nested loop join)

Implementing: Joins

Solution 6 (Index-Nested-Loop)

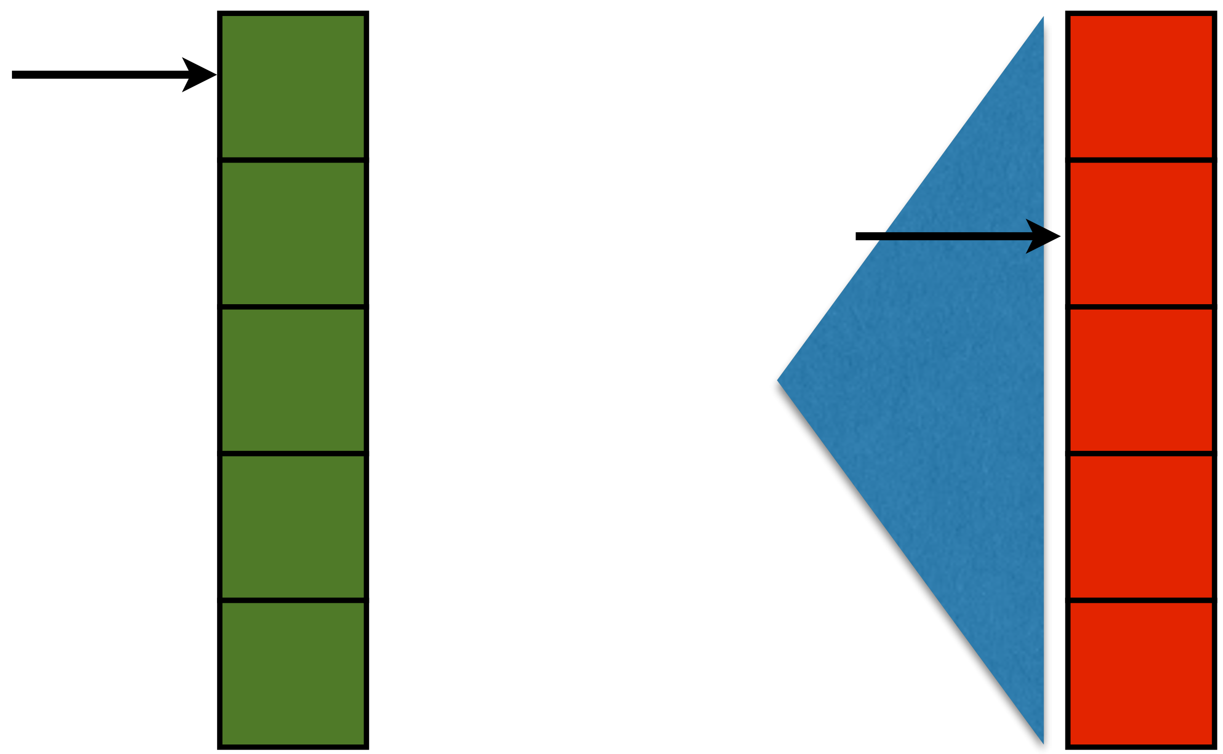
Like nested-loop, but use an index to make the inner loop much faster!



Implementing: Joins

Solution 6 (Index-Nested-Loop)

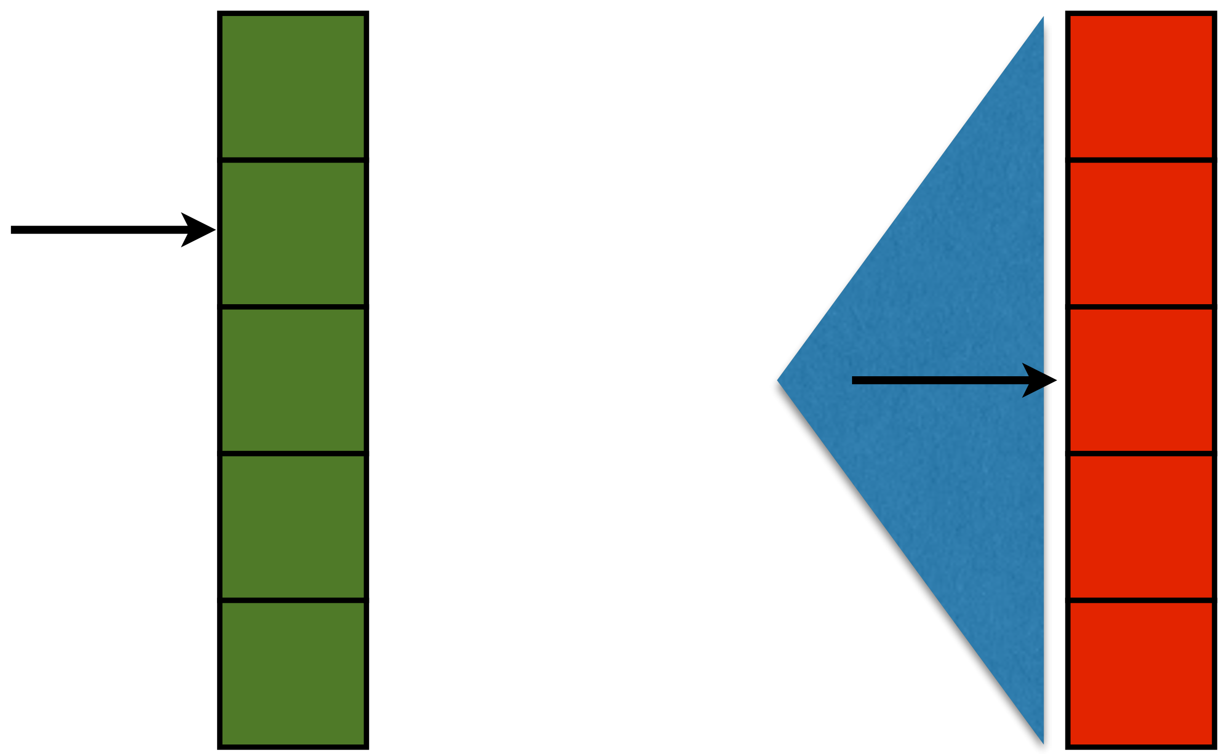
Like nested-loop, but use an index to make the inner loop much faster!



Implementing: Joins

Solution 6 (Index-Nested-Loop)

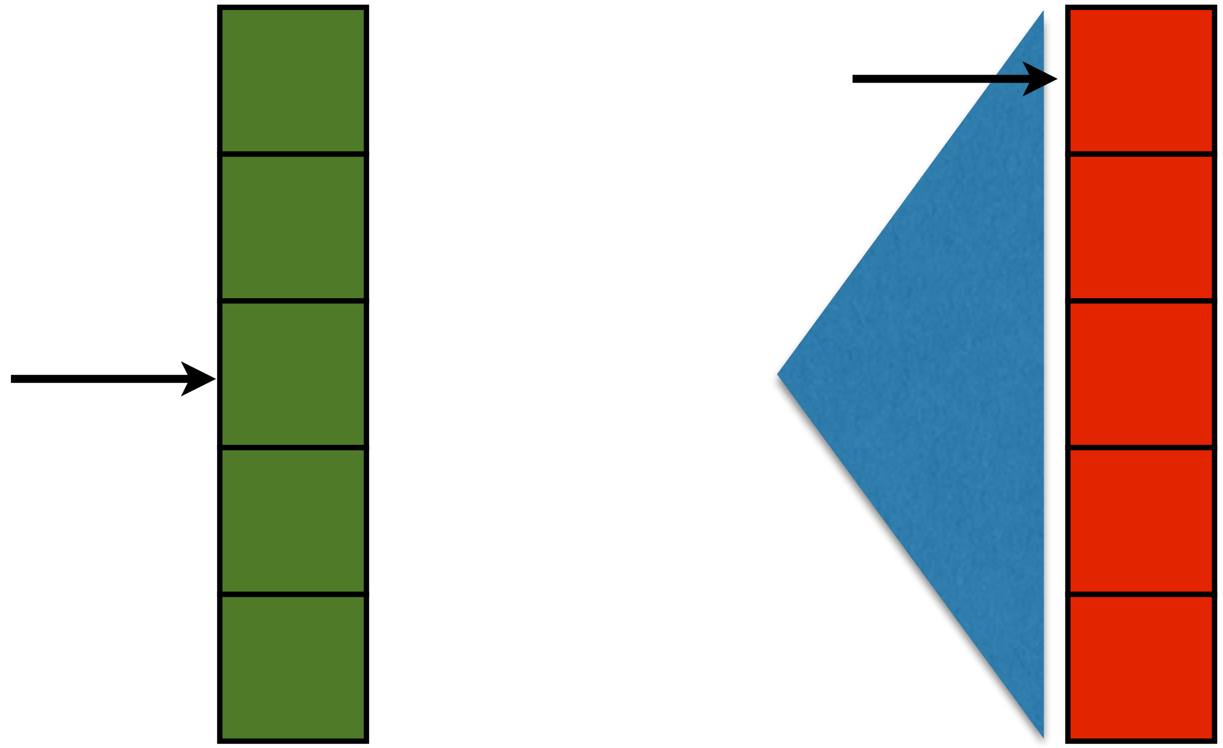
Like nested-loop, but use an index to make the inner loop much faster!



Implementing: Joins

Solution 6 (Index-Nested-Loop)

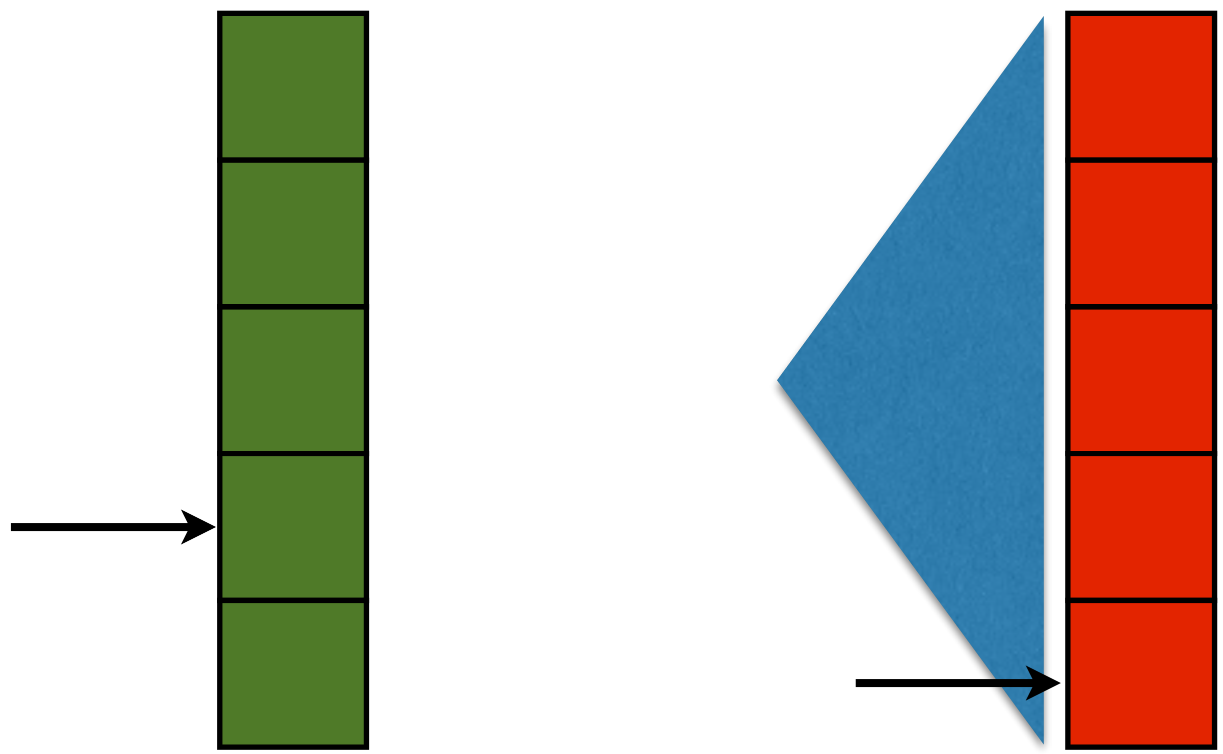
Like nested-loop, but use an index to make the inner loop much faster!



Implementing: Joins

Solution 6 (Index-Nested-Loop)

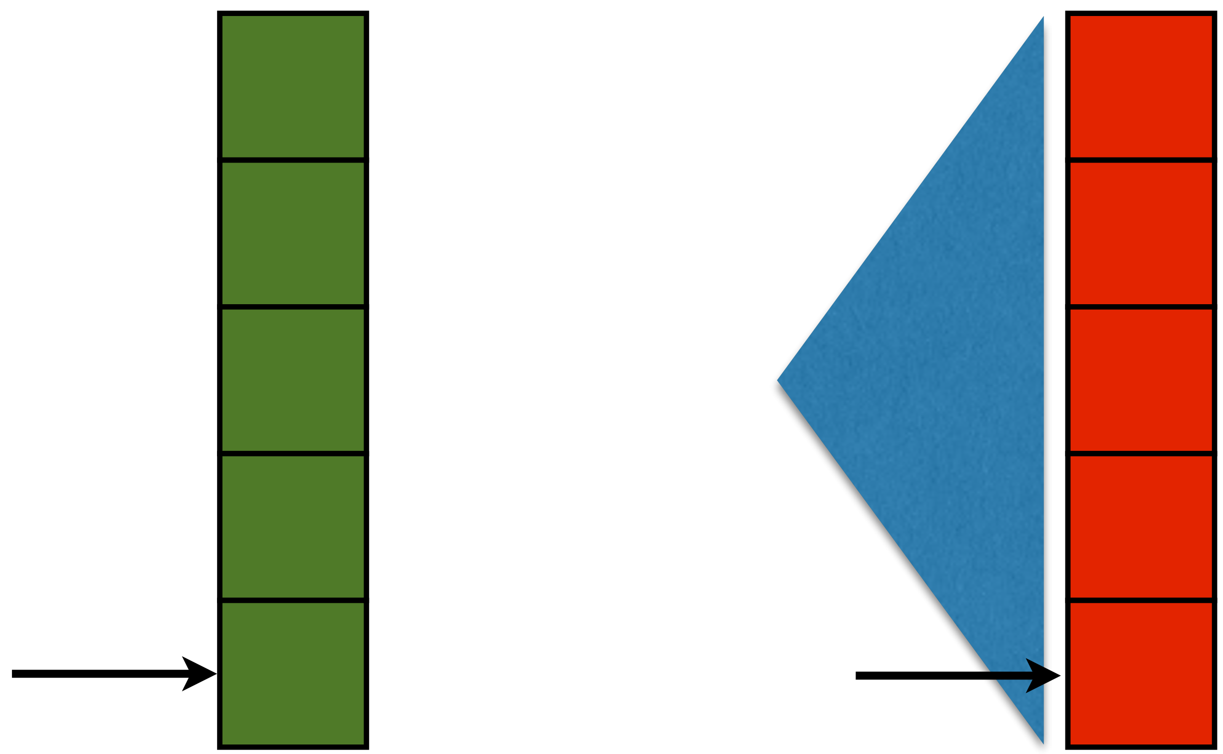
Like nested-loop, but use an index to make the inner loop much faster!



Implementing: Joins

Solution 6 (Index-Nested-Loop)

Like nested-loop, but use an index to make the inner loop much faster!



What are the tradeoffs of each algorithm?

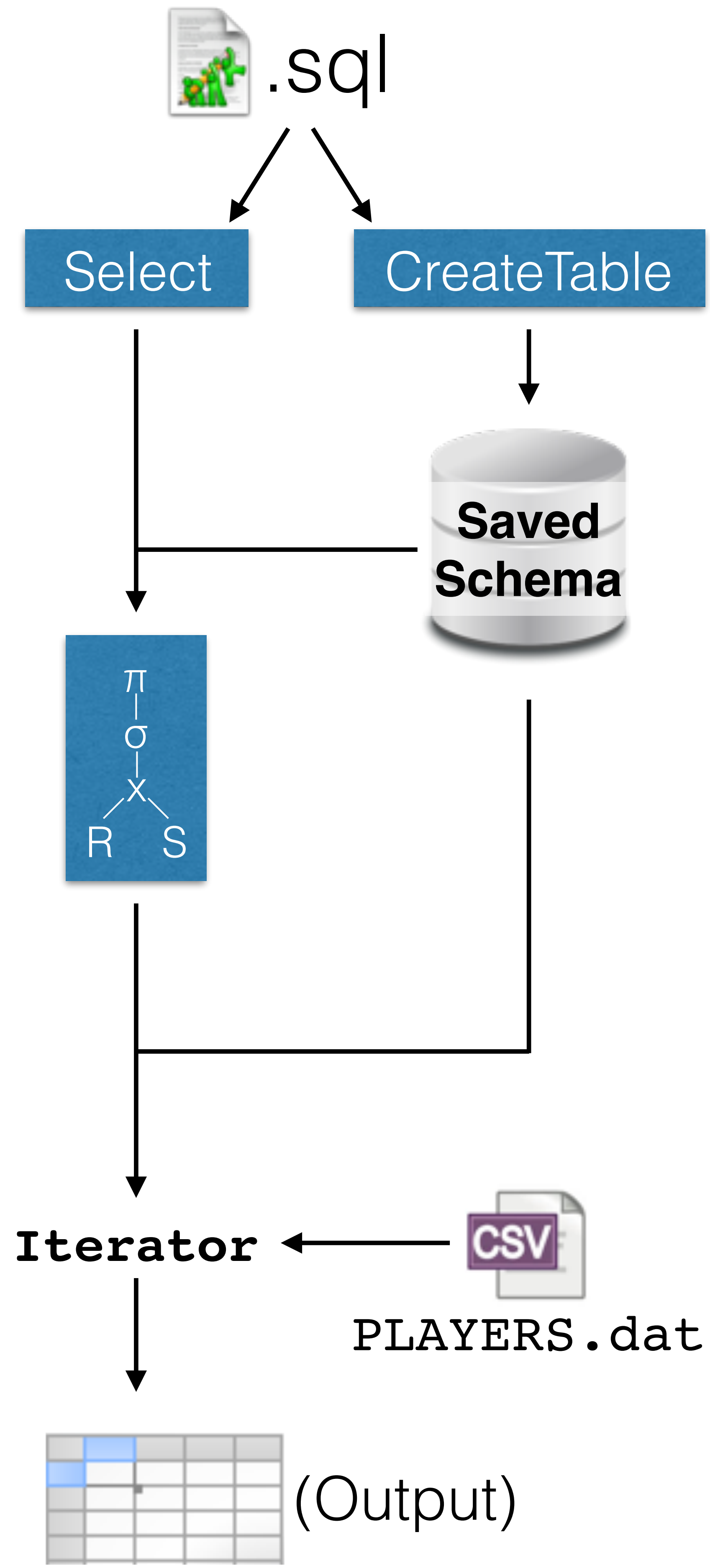
What properties do we care about?

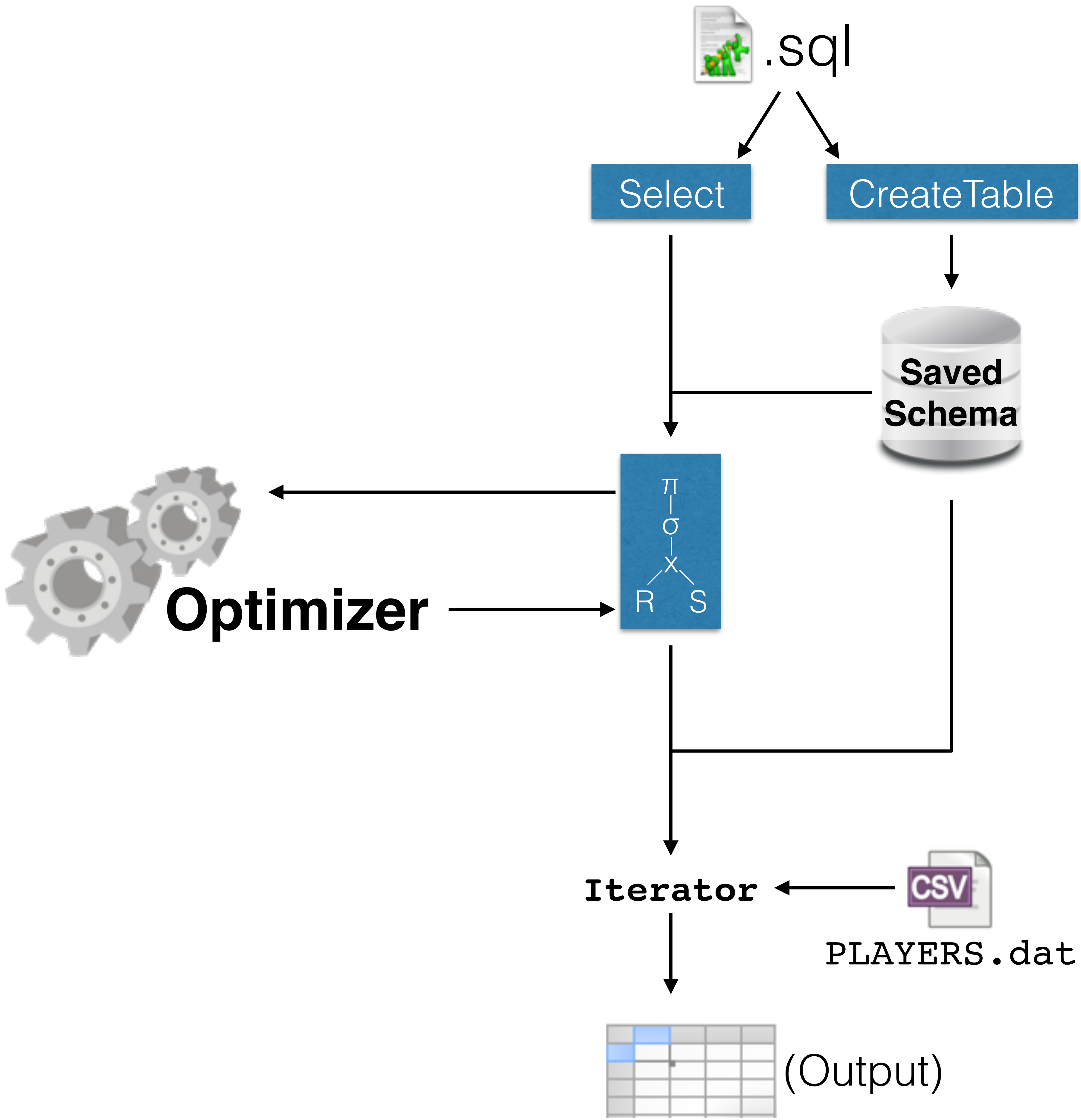
How do the algorithms compare?

Implementing: Joins

Tradeoffs

	Pipelined?	Memory Requirements?	Predicate Limitation?
Nested Loop	1/2	1 Table	No
Block-Nested Loop	No	2 ‘Blocks’	No
Index-Nested Loop	1/2	1 Tuple (+Index)	Single Comparison
Sort-Merge	If Data Sorted	Same as reqs. of Sorting Inputs	Equality Only
Hash	No	Max of 1 Page per Bucket and All Pages in Any Bucket	Equality Only
Grace Hash	1/2	Hash Table	Equality Only


.sql
Select
CreateTable
Saved Schema
π
σ
X
R
S
Iterator
CSV
PLAYERS.dat
(Output)



.sql
Select
CreateTable
Saved Schema
π
σ
X
R
S
Optimizer
Iterator
CSV
PLAYERS.dat
(Output)

Equivalent Expressions

They look the same, but one is good, one is evil

≠

=

Two different expressions of the “same” character

Query Optimization

If X and Y are <u>equivalent</u> and Y is <u>better</u>…

… then replace all Xs with Ys

Equivalent Expressions

R

< A >
< 1 >
< 2 >
< 2 >

S

< A, B >
< 2, 4 >
< 3, 5 >
< 3, 6 >

Equivalent Expressions

R

< A >
< 1 >
< 2 >
< 2 >

S

< A, B >
< 2, 4 >
< 3, 5 >
< 3, 6 >

Is $R = S$ **?**

Equivalent Expressions

R

< A >
< 1 >
< 2 >
< 2 >

S

< A, B >
< 2, 4 >
< 3, 5 >
< 3, 6 >

Is $R = S$ **?**

Is $R = \pi_A(S)$ **?**

Equivalent Expressions

R
< A >
< 1 >
< 2 >
< 2 >

S
< A, B >
< 2, 4 >
< 3, 5 >
< 3, 6 >

Is $R = S$ **?**

Is $R = \pi_A(S)$ **?**

Is $R = \pi_{A \leftarrow (A-1)}(S)$ **?**

Equivalent Expressions

R

```
< A >
-----
< 1 >
< 2 >
< 2 >
```

S

```
< A, B >
--------
< 2, 4 >
< 3, 5 >
< 3, 6 >
```

Is $R = S$ **?**

Is $R = \pi_A(S)$ **?**

Is $R = \pi_{A \leftarrow (A-1)}(S)$ **?**

Is $\pi_{A \leftarrow (A+1)}(R) = \pi_A(S)$ **?**

Equivalent Expressions

Two expressions are equivalent
if they produce the same output

Equivalent Expressions

Two expressions are equivalent
if they produce the same output

but…

Equivalent Expressions

< A >
< 1 >
< 2 >
< 2 >

=?

< A >
< 2 >
< 1 >
< 2 >

=?

< A >
< 1 >
< 2 >

Equivalence under…

- **Bag Semantics**: The same tuples (order-independent)
- **Set Semantics**: The same set of tuples (count-independent)
- **List Semantics**: The same tuples (order matters)

Equivalent Expressions

< A >
< 1 >
< 2 >
< 2 >

=?

< A >
< 2 >
< 1 >
< 2 >

=?

< A >
< 1 >
< 2 >

Equivalence under…

- **Bag Semantics**: The same tuples (order-independent)
- **Set Semantics**: The same set of tuples (count-independent)
- **List Semantics**: The same tuples (order matters)

RA Equivalencies

Selection

$$\sigma_{c_1 \wedge c_2}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(R))$$ (Decomposable)

$$\sigma_{c_1 \vee c_2}(R) \equiv \delta(\sigma_{c_1}(R) \cup \sigma_{c_2}(R))$$ (Decomposable)

$$\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R))$$ (Commutative)

Projection

$$\pi_a(R) \equiv \pi_a(\pi_{a \cup b}(R))$$ (Idempotent)

Cross Product (and Join)

$$R \times (S \times T) \equiv (R \times S) \times T$$ (Associative)

$$(R \times S) \equiv (S \times R)$$ (Commutative)

Try It: Show that $R \times (S \times T) \equiv T \times (R \times S)$

Selection and Projection

$$\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$$

Selection <u>commutes</u> with Projection
(but only if attribute set **a** and condition **c** are *compatible*)

a must include all columns referenced by **c**

<u>Show that</u>

$$\pi_a(\sigma_c(R)) \equiv \pi_a(\sigma_c(\pi_{a \cup \mathtt{cols}(c)}(R)))$$

Selection and Projection

$$\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$$

Selection <u>commutes</u> with Projection
(but only if attribute set **a** and condition **c** are *compatible*)

a must include all columns referenced by **c**

<u>Show that</u>

$$\pi_a(\sigma_c(R)) \equiv \pi_a(\sigma_c(\pi_{a \cup \mathtt{cols}(c)}(R)))$$

When is this rewrite a good idea?

Join

$$\sigma_c(R \times S) \equiv R \bowtie_c S$$

Selection <u>combines</u> with Cross Product
to form a Join as per the definition of Join
(Note: This only helps if we have a join algorithm for conditions like **c**)

<u>Show that</u>

$$\sigma_{(R.B=S.B)\wedge(R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R \bowtie_{(R.B=S.B)} S)$$

Join

$$\sigma_c(R \times S) \equiv R \bowtie_c S$$

Selection combines with Cross Product
to form a Join as per the definition of Join
(Note: This only helps if we have a join algorithm for conditions like **c**)

Show that

$$\sigma_{(R.B=S.B)\wedge(R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R \bowtie_{(R.B=S.B)} S)$$

When is this rewrite a good idea?

Selection and Cross Product

$$\sigma_c(R \times S) \equiv (\sigma_c(R) \times S)$$

Selection <u>commutes</u> with Cross Product
(but only if condition **c** references attributes of R exclusively)

<u>Show that</u>

$$\sigma_{(R.B=S.B)\wedge(R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R) \bowtie_{(R.B=S.B)} S$$

Selection and Cross Product

$$\sigma_c(R \times S) \equiv (\sigma_c(R) \times S)$$

Selection commutes with Cross Product
(but only if condition **c** references attributes of R exclusively)

Show that

$$\sigma_{(R.B=S.B)\wedge(R.A>3)}(R \times S) \equiv \sigma_{(R.A>3)}(R) \bowtie_{(R.B=S.B)} S$$

When is this rewrite a good idea?

Projection and Cross Product

$$\pi_a(R \times S) \equiv (\pi_{a_1}(R)) \times (\pi_{a_2}(S))$$

Projection <u>commutes</u> (distributes) over Cross Product
(where $\mathbf{a_1}$ and $\mathbf{a_2}$ are the attributes in $\mathbf{a}$ from R and S respectively)

<u>Show that</u>

$$\pi_a(R \bowtie_c S) \equiv (\pi_{a_1}(R)) \bowtie_c (\pi_{a_2}(S))$$

(under what condition)

How can we work around this limitation?

$$\pi_a((\pi_{a_1 \cup (\mathtt{cols}(c) \cap \mathtt{cols}(R))}(R)) \bowtie_c (\pi_{a_2 \cup (\mathtt{cols}(c) \cap \mathtt{cols}(S))}(S)))$$

Projection and Cross Product

$$\pi_a(R \times S) \equiv (\pi_{a_1}(R)) \times (\pi_{a_2}(S))$$

Projection commutes (distributes) over Cross Product
(where $\mathbf{a_1}$ and $\mathbf{a_2}$ are the attributes in $\mathbf{a}$ from R and S respectively)

Show that

$$\pi_a(R \bowtie_c S) \equiv (\pi_{a_1}(R)) \bowtie_c (\pi_{a_2}(S))$$

(under what condition)

How can we work around this limitation?

$$\pi_a((\pi_{a_1 \cup (\mathtt{cols}(c) \cap \mathtt{cols}(R))}(R)) \bowtie_c (\pi_{a_2 \cup (\mathtt{cols}(c) \cap \mathtt{cols}(S))}(S)))$$

When is this rewrite a good idea?

RA Equivalencies

Union and Intersections are <u>Commutative</u> and <u>Associative</u>

Selection and Projection both commute with both Union and Intersection

RA Equivalencies

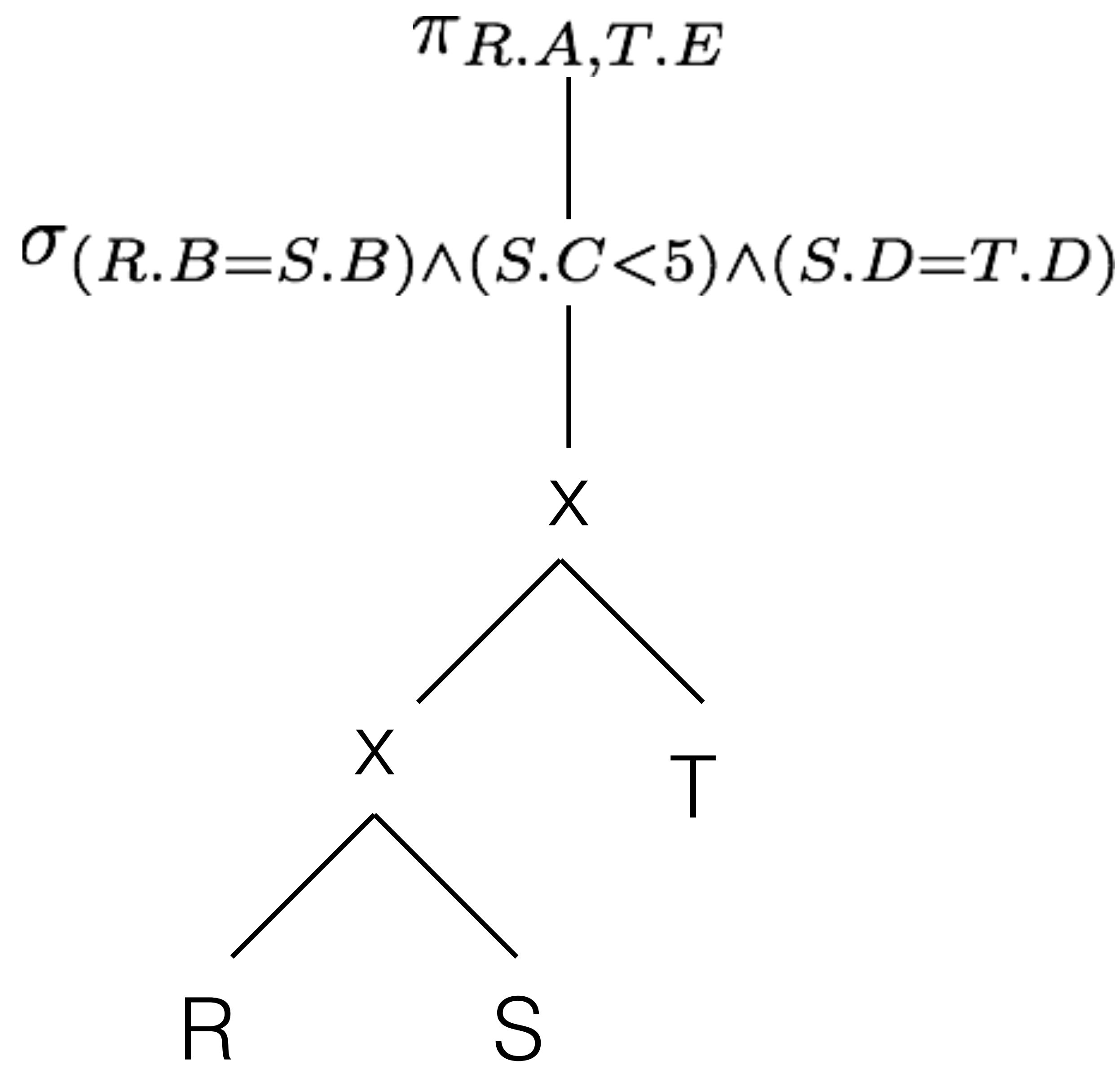
Union and Intersections are <u>Commutative</u> and <u>Associative</u>

Selection and Projection both commute with both Union and Intersection

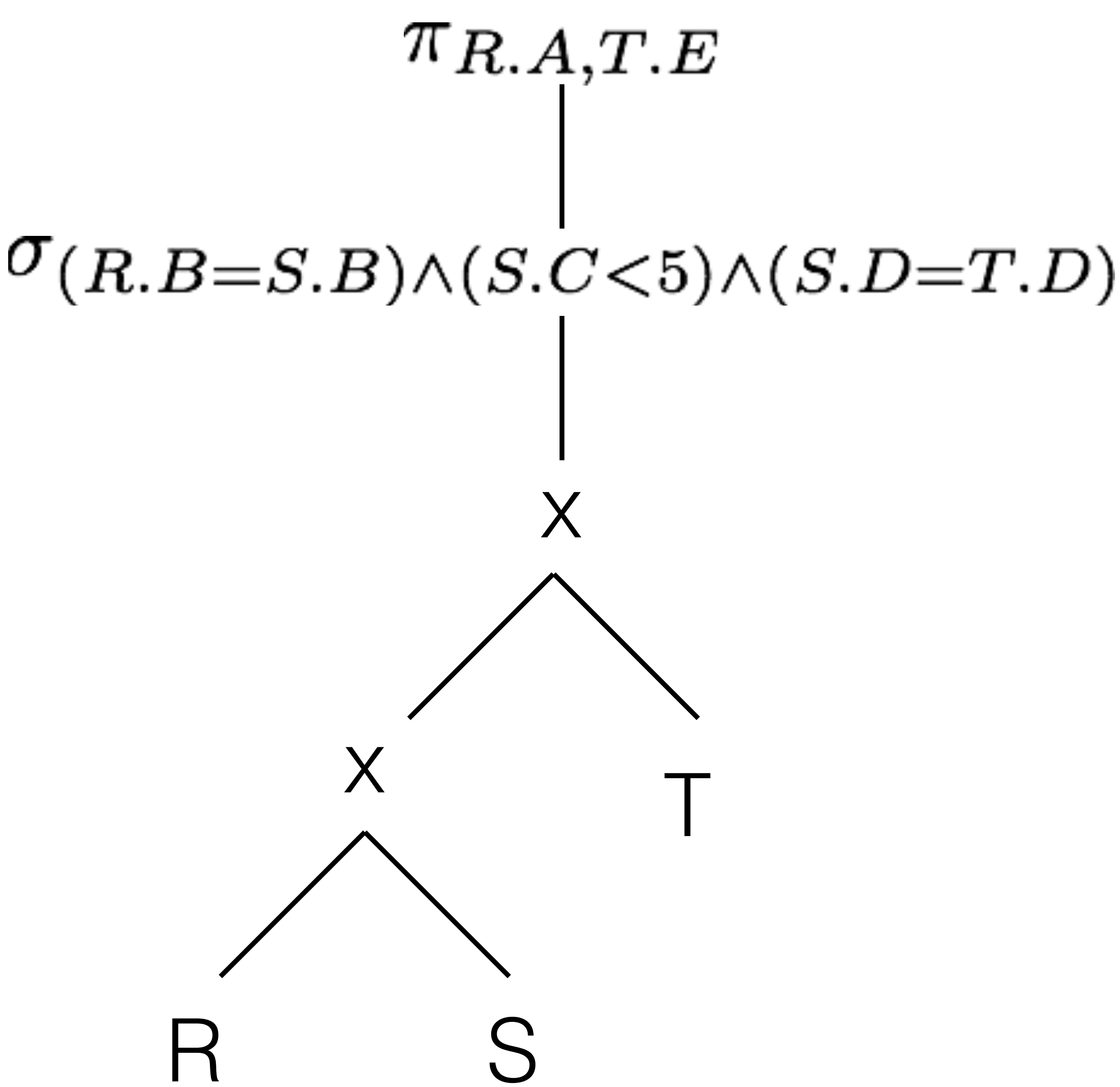
When is this rewrite a good idea?

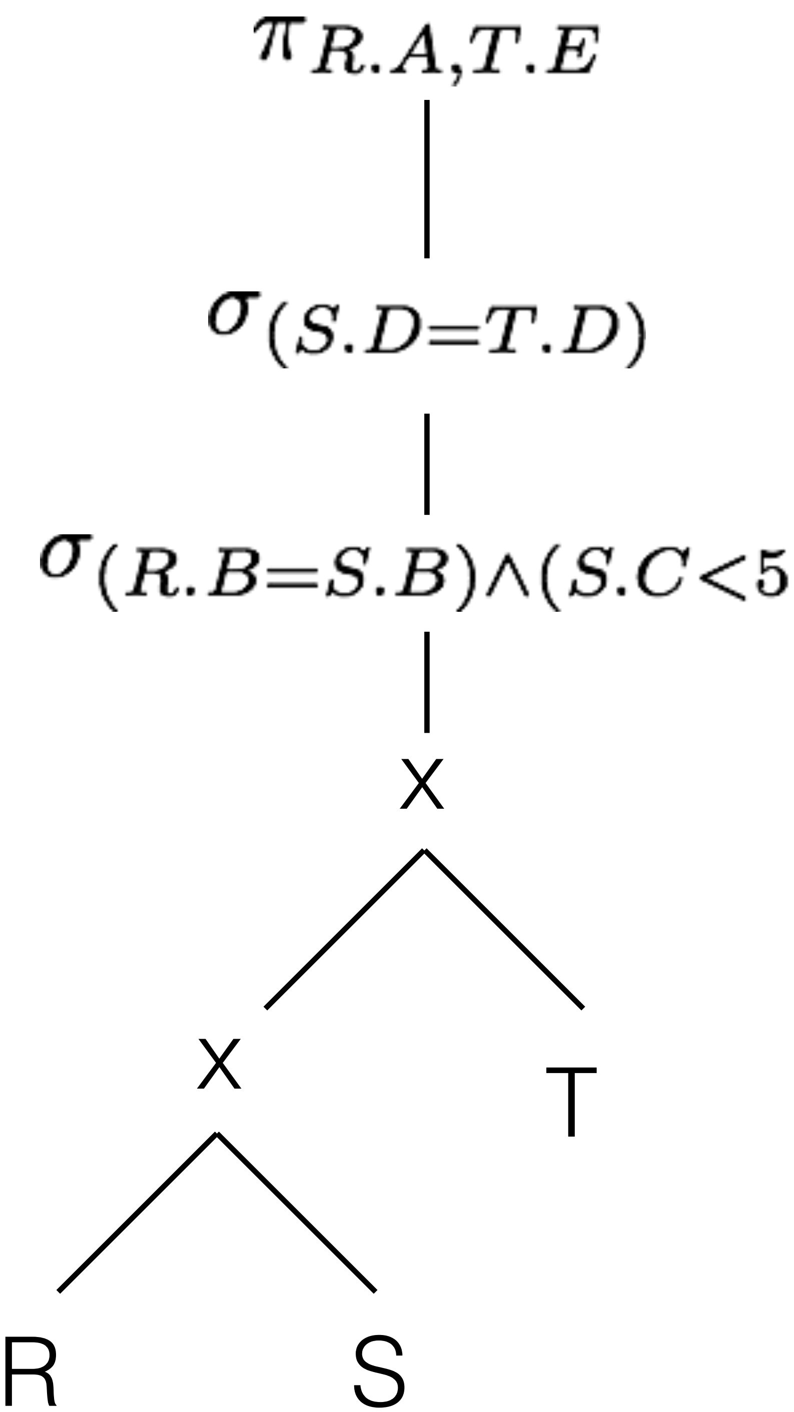
Example

```
SELECT R.A, T.E
  FROM R, S, T
 WHERE R.B = S.B
   AND S.C < 5
   AND S.D = T.D
```

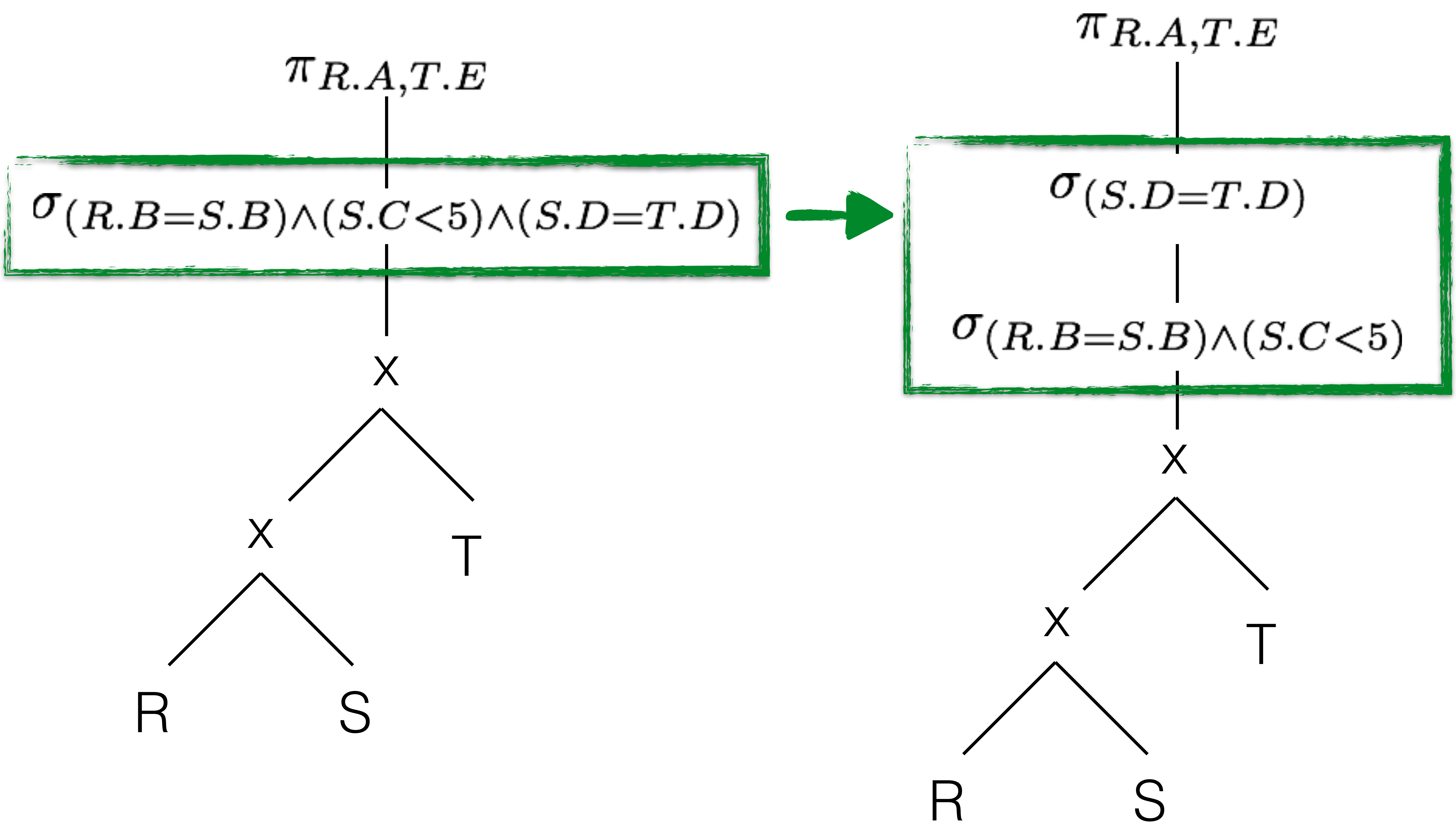


Example


$\pi_{R.A,T.E}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)\wedge(S.D=T.D)}$
X
X
T
R
S

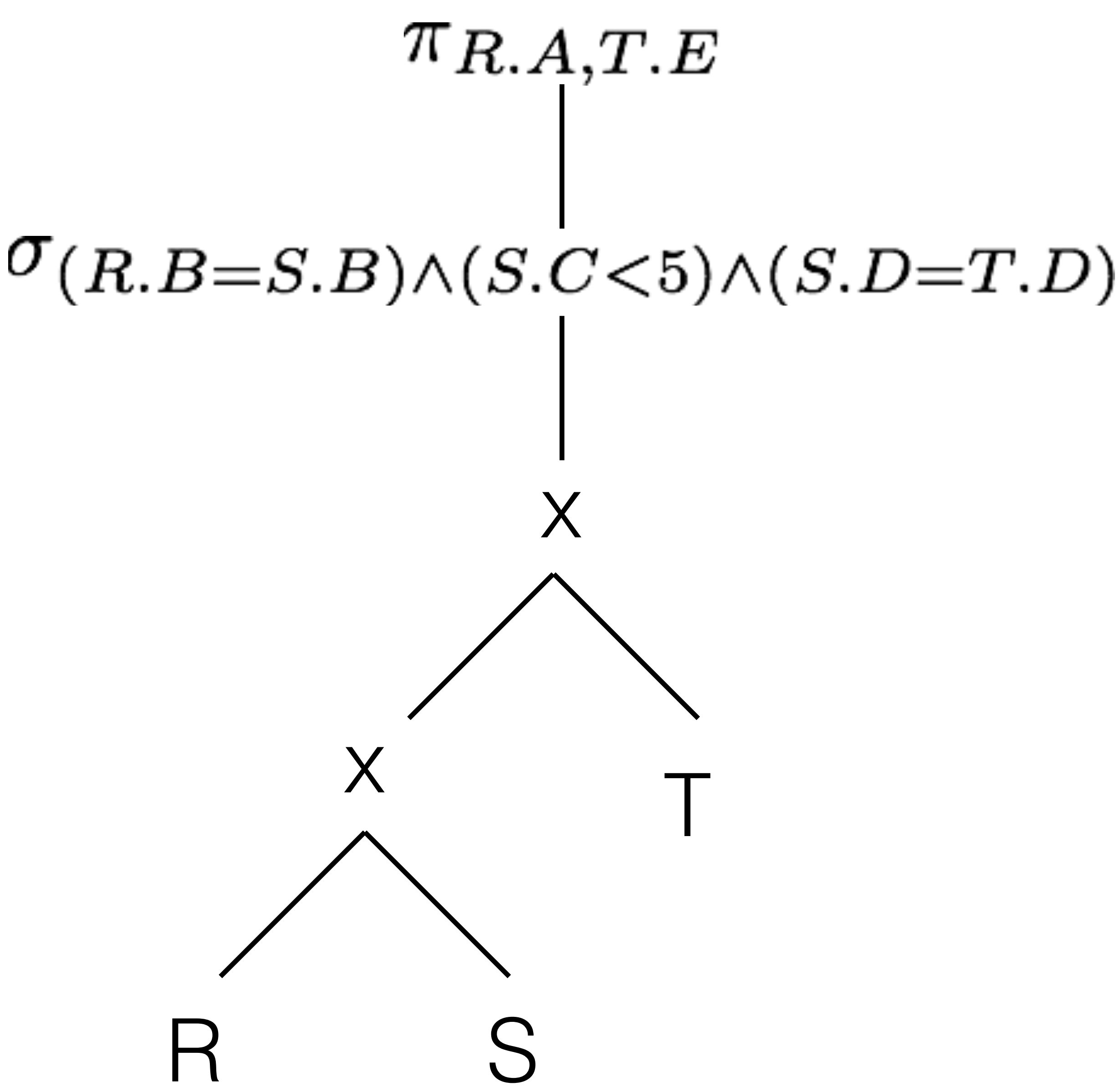

$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
X
X
T
R
S

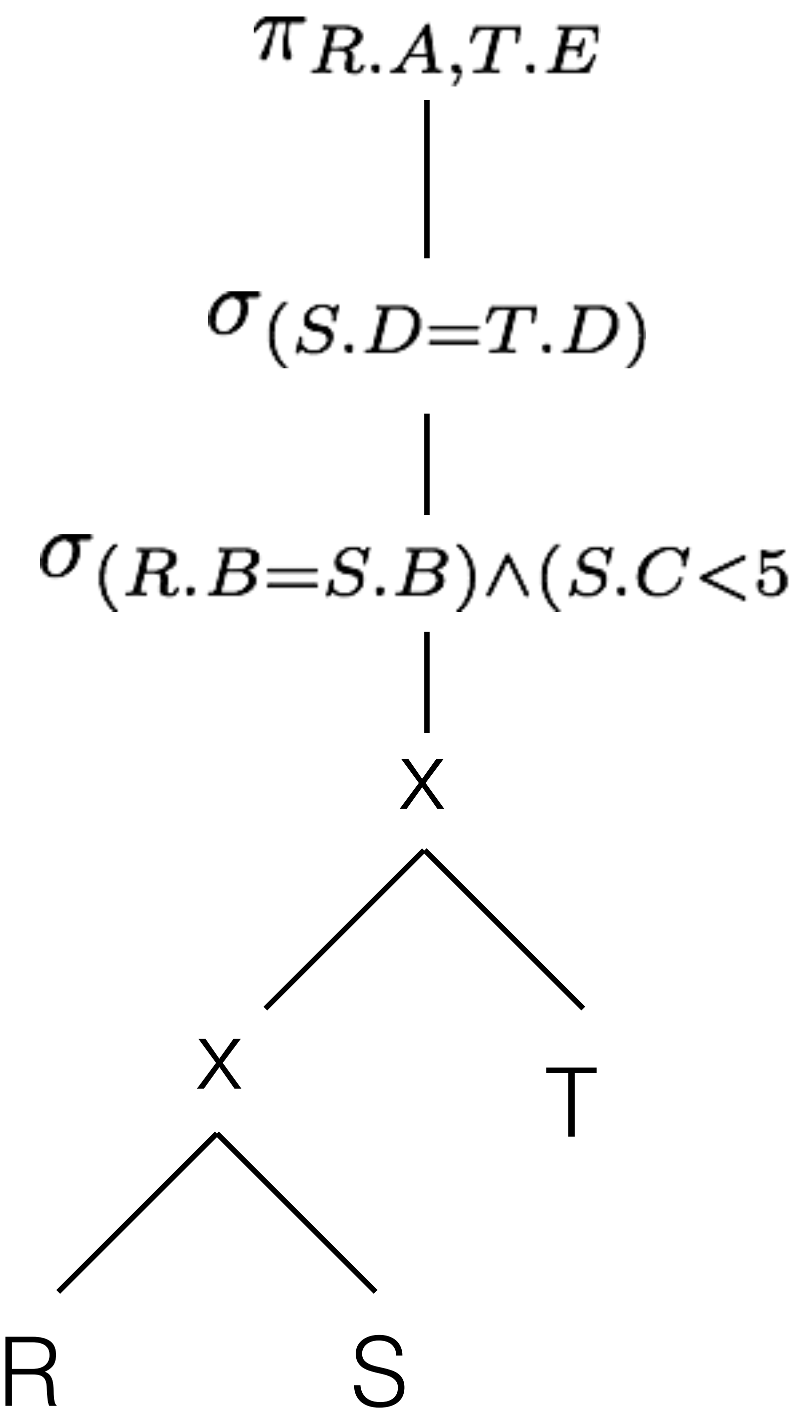
Example



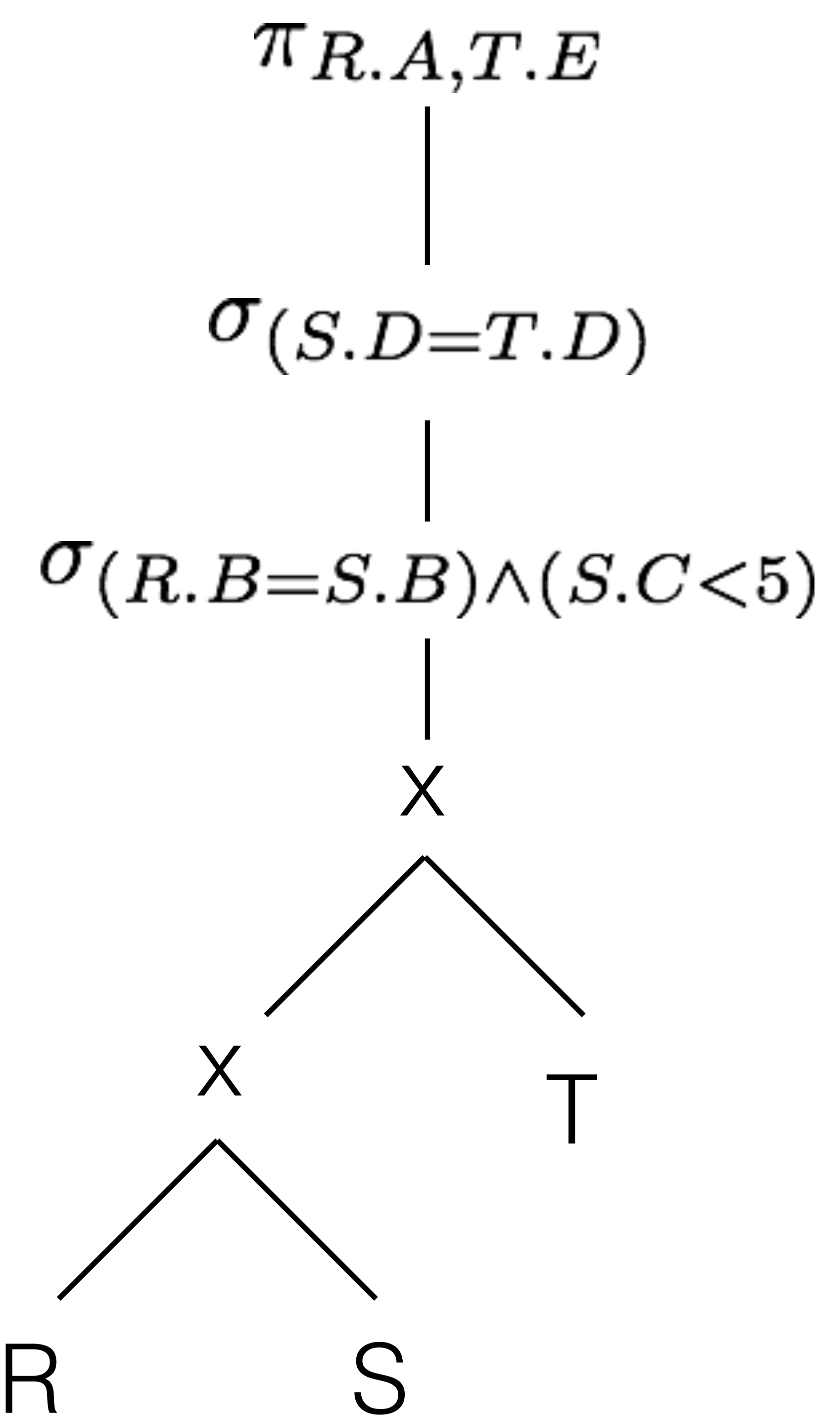
$\pi_{R.A,T.E}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)\wedge(S.D=T.D)}$
X
X
T
R
S
$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
X
X
T
R
S

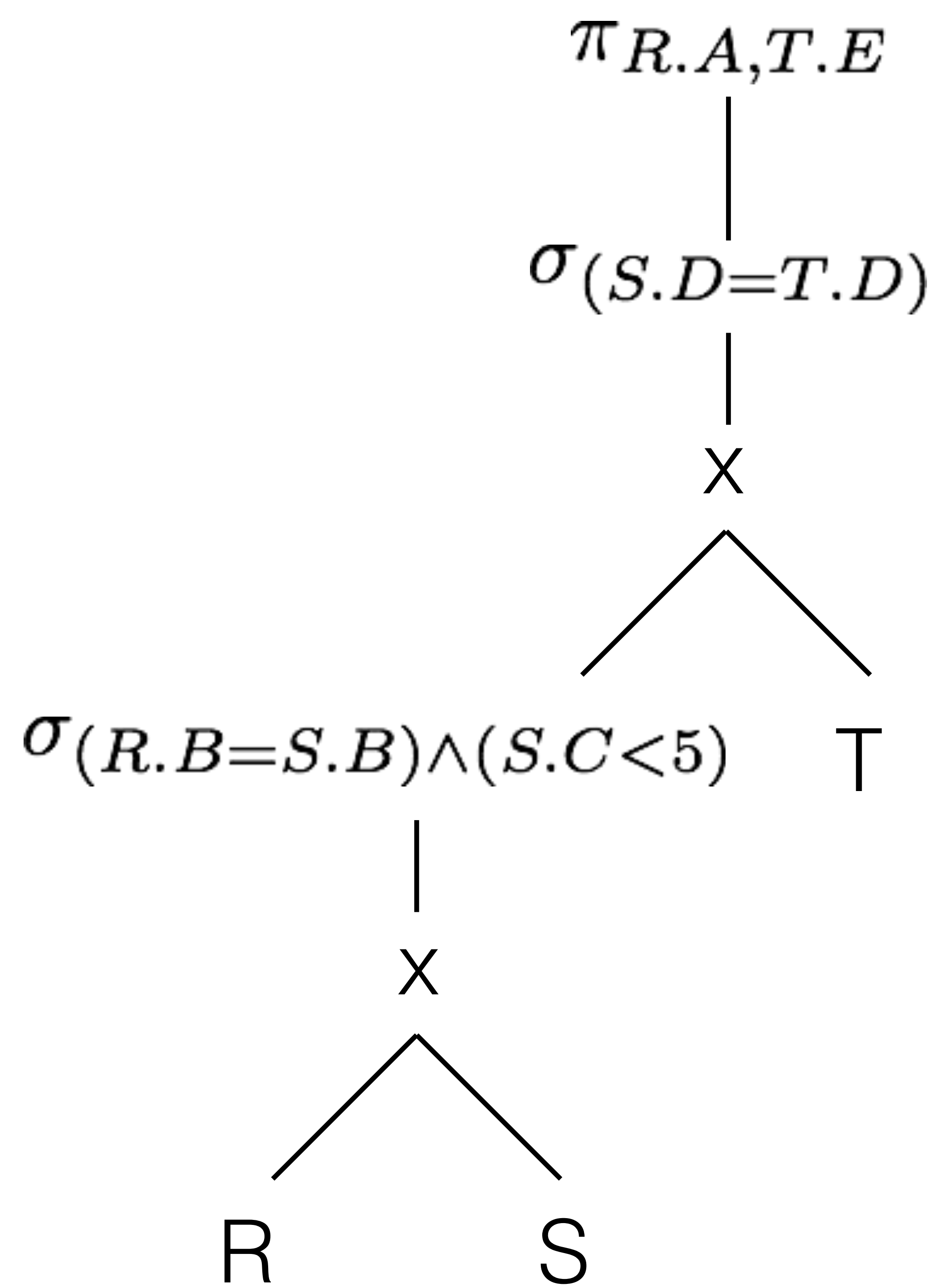
Example


$\pi_{R.A,T.E}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)\wedge(S.D=T.D)}$
X
X
T
R
S

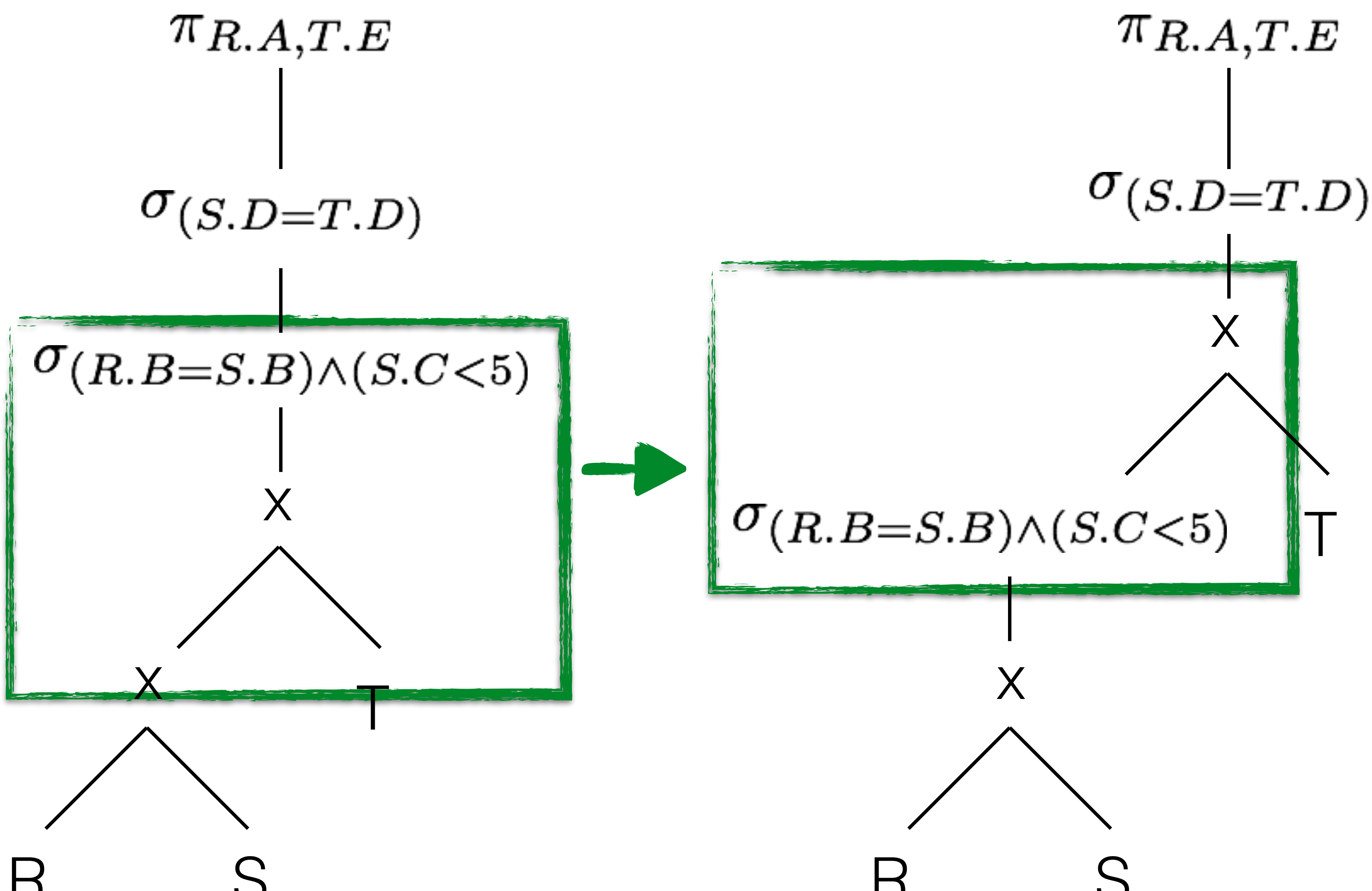

$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
X
X
T
R
S

Example


$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
X
X
T
R
S

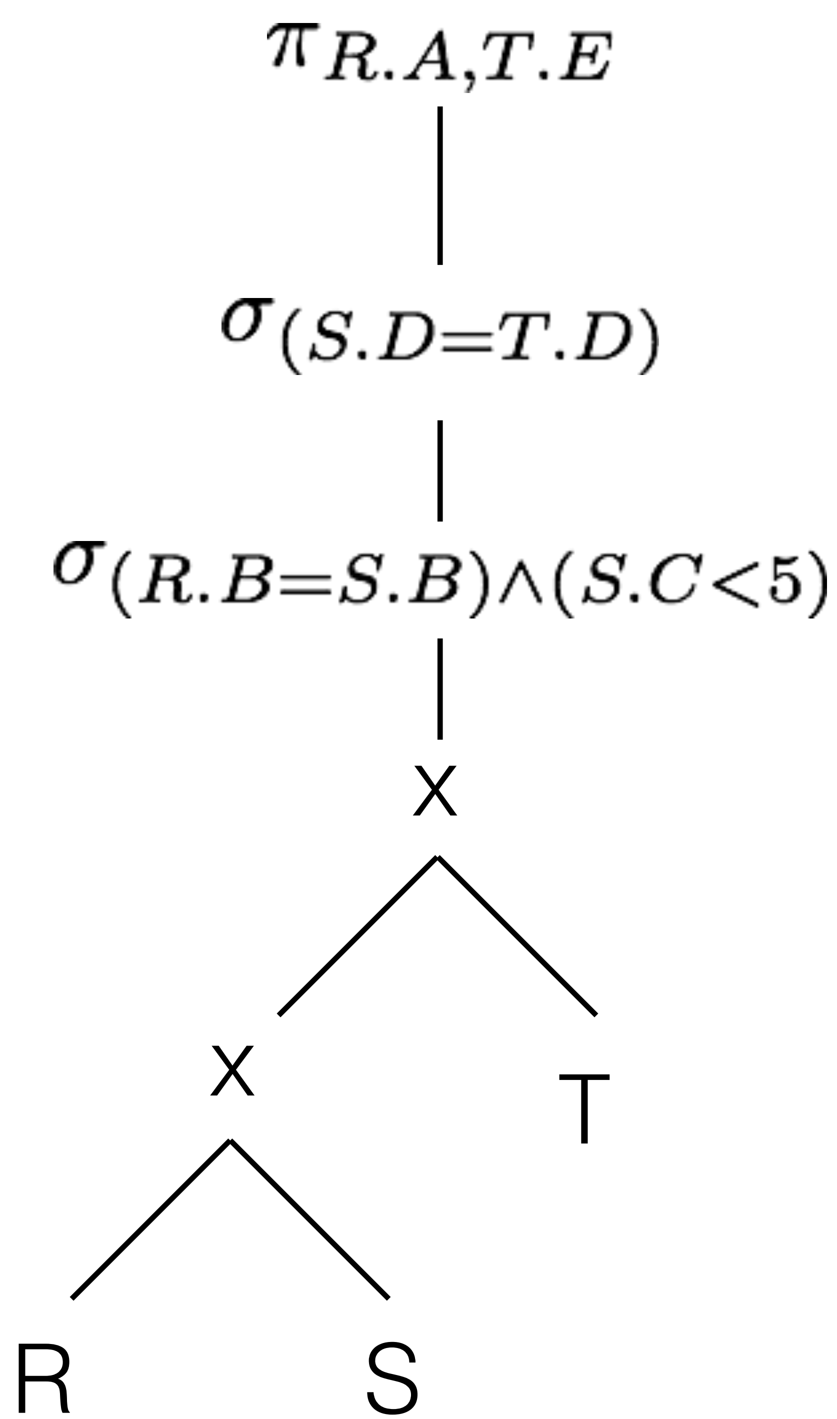

$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
X
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
X
R
S

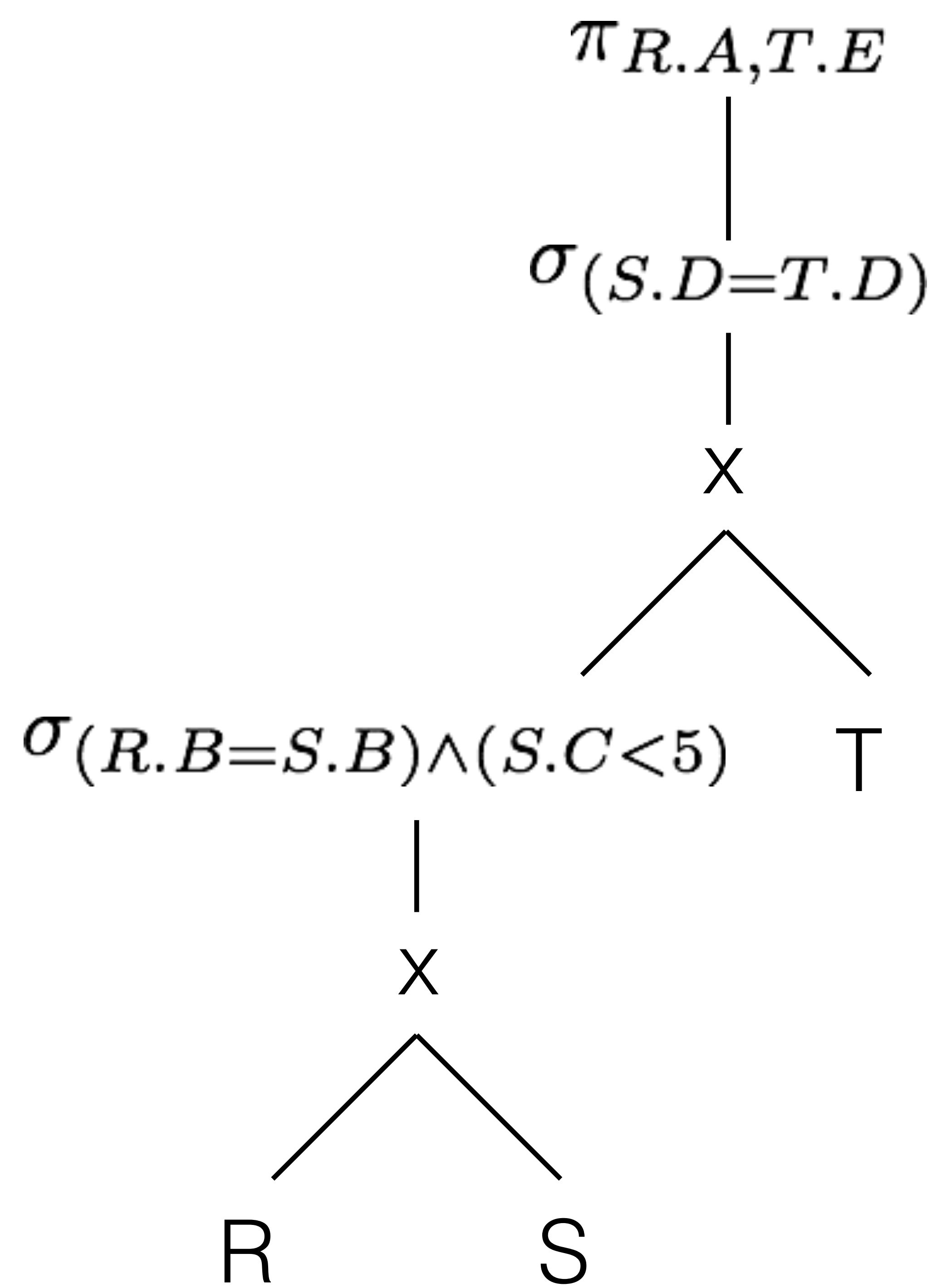
Example



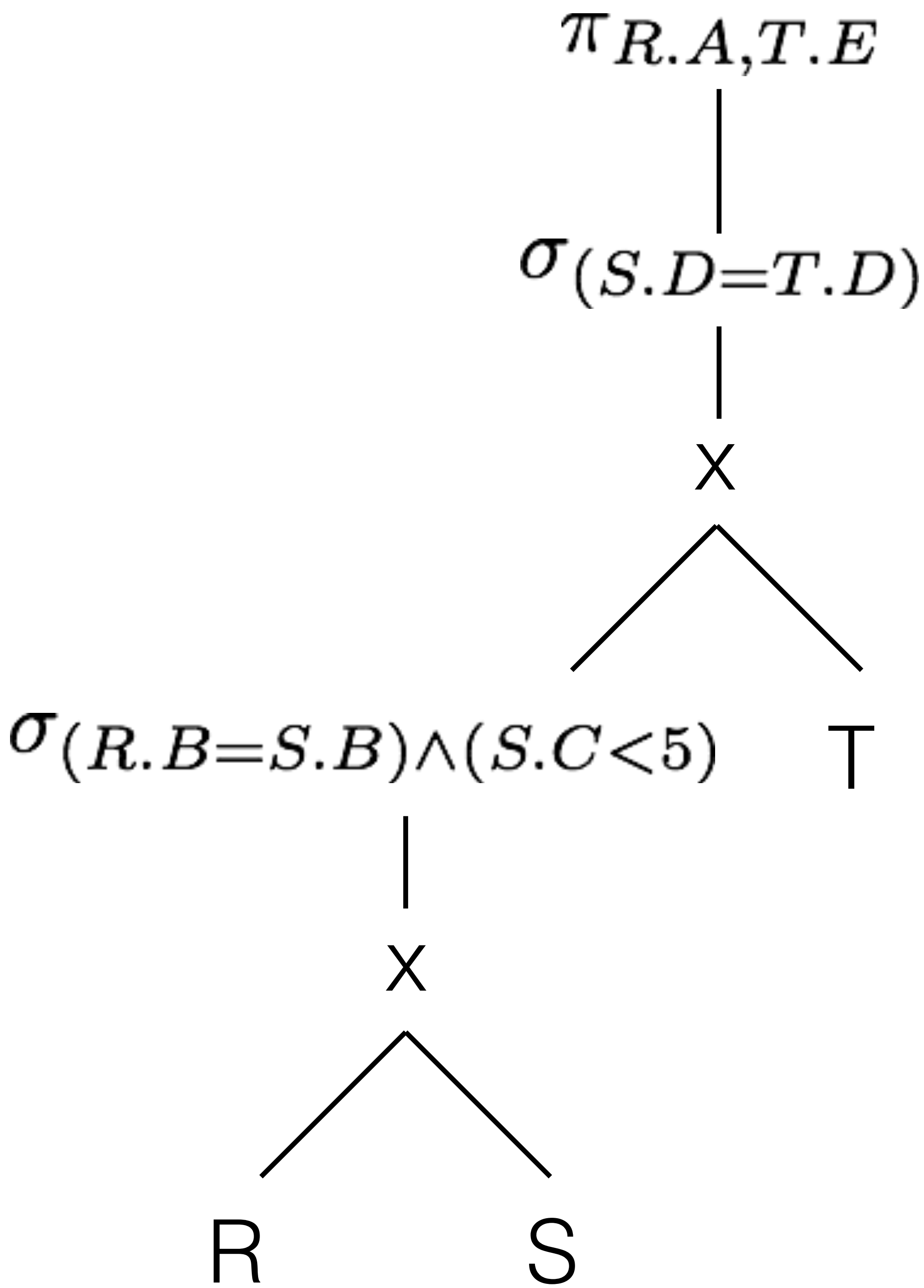
$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
X
X
T
R
S
$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
X
T
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
X
R
S

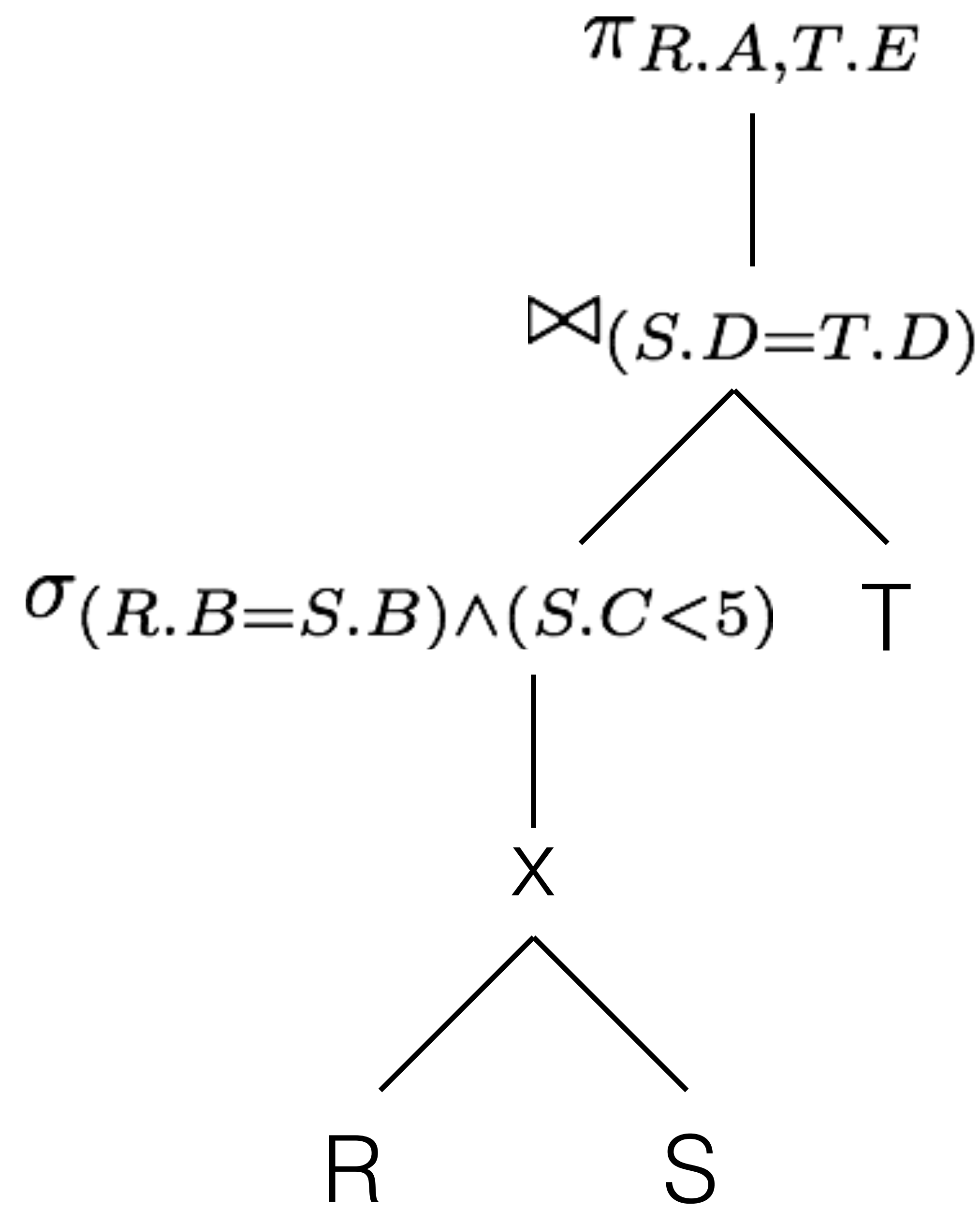
Example


$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
X
X
T
R
S

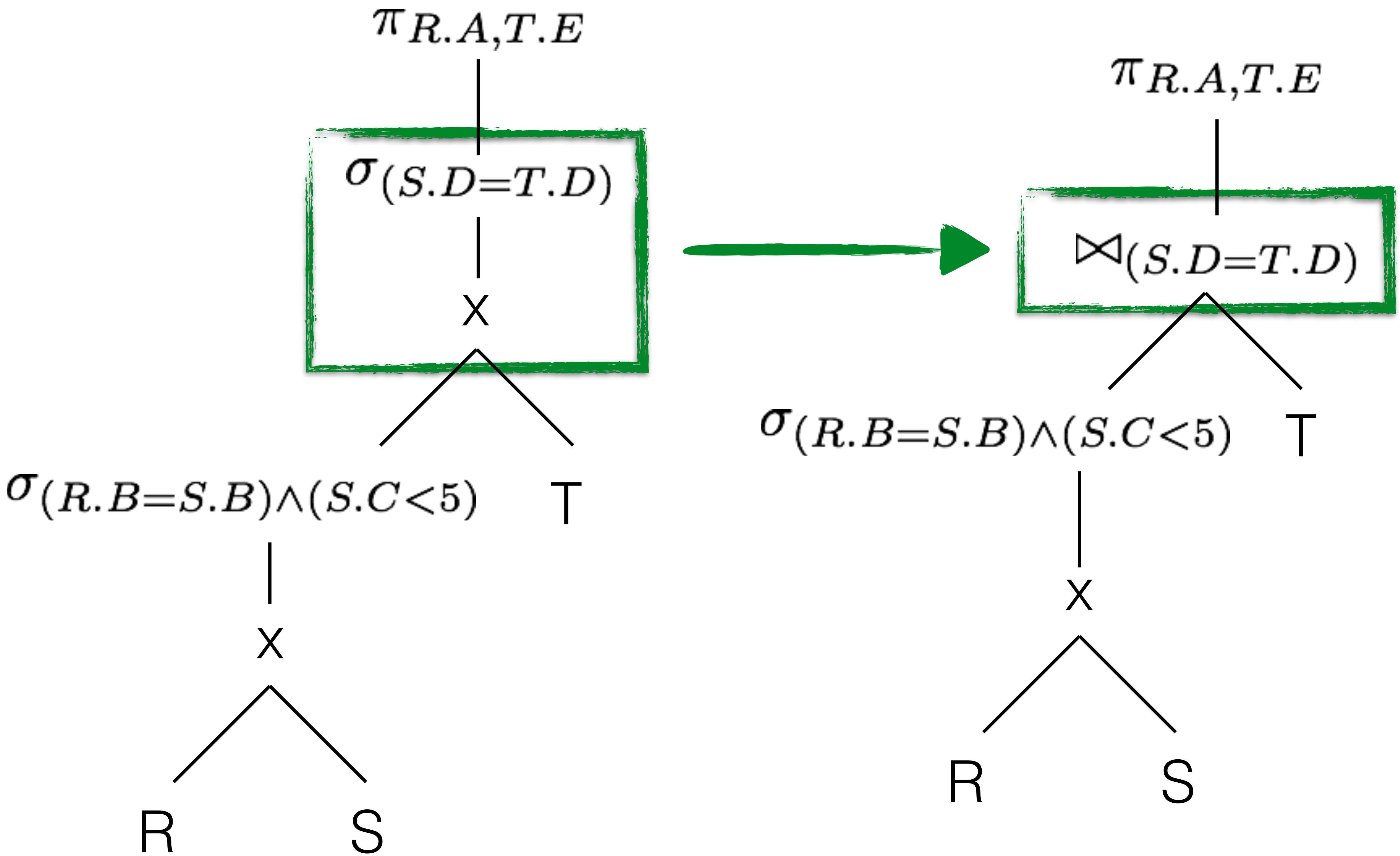

$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
X
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
X
R
S

Example


$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
x
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
x
R
S

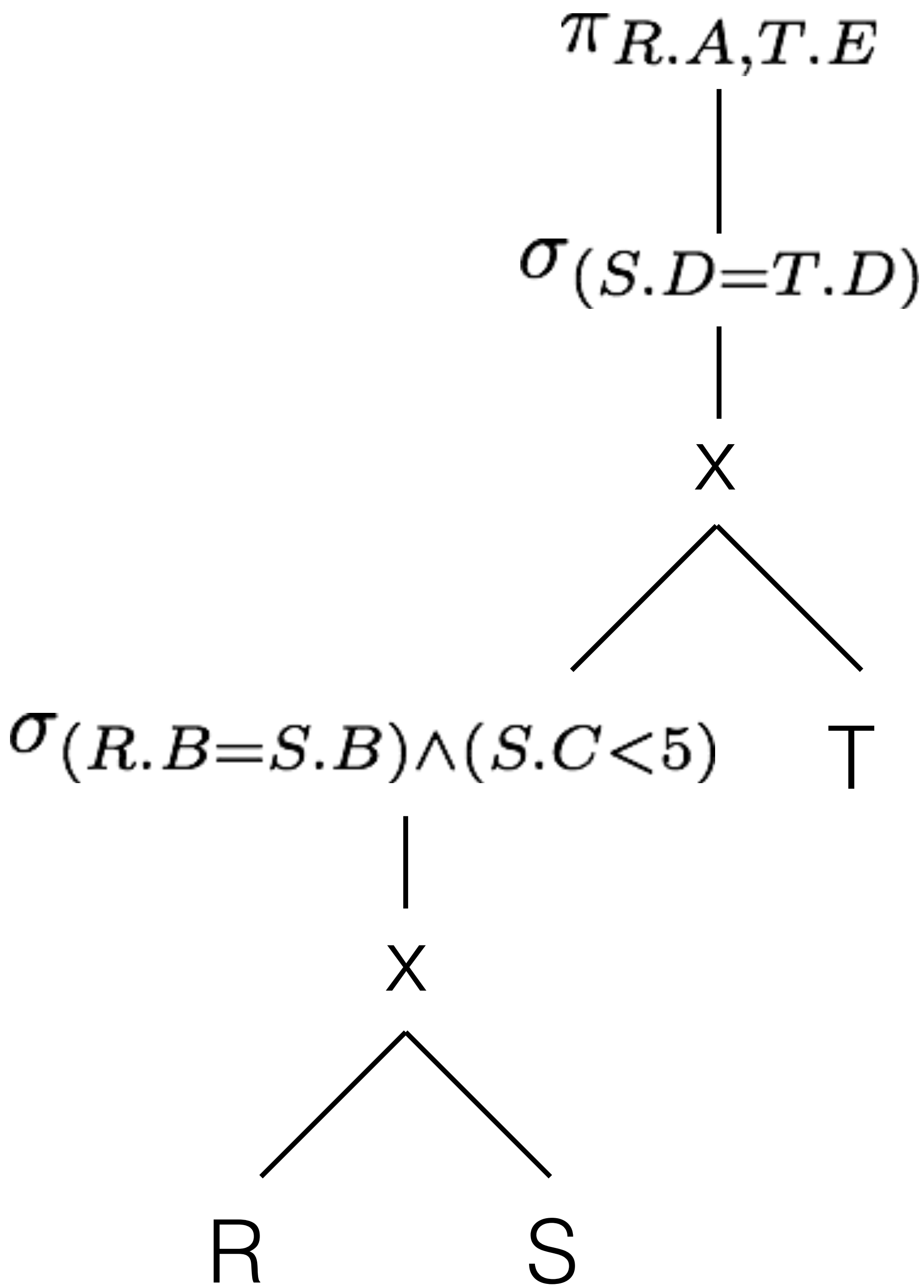

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
x
R
S

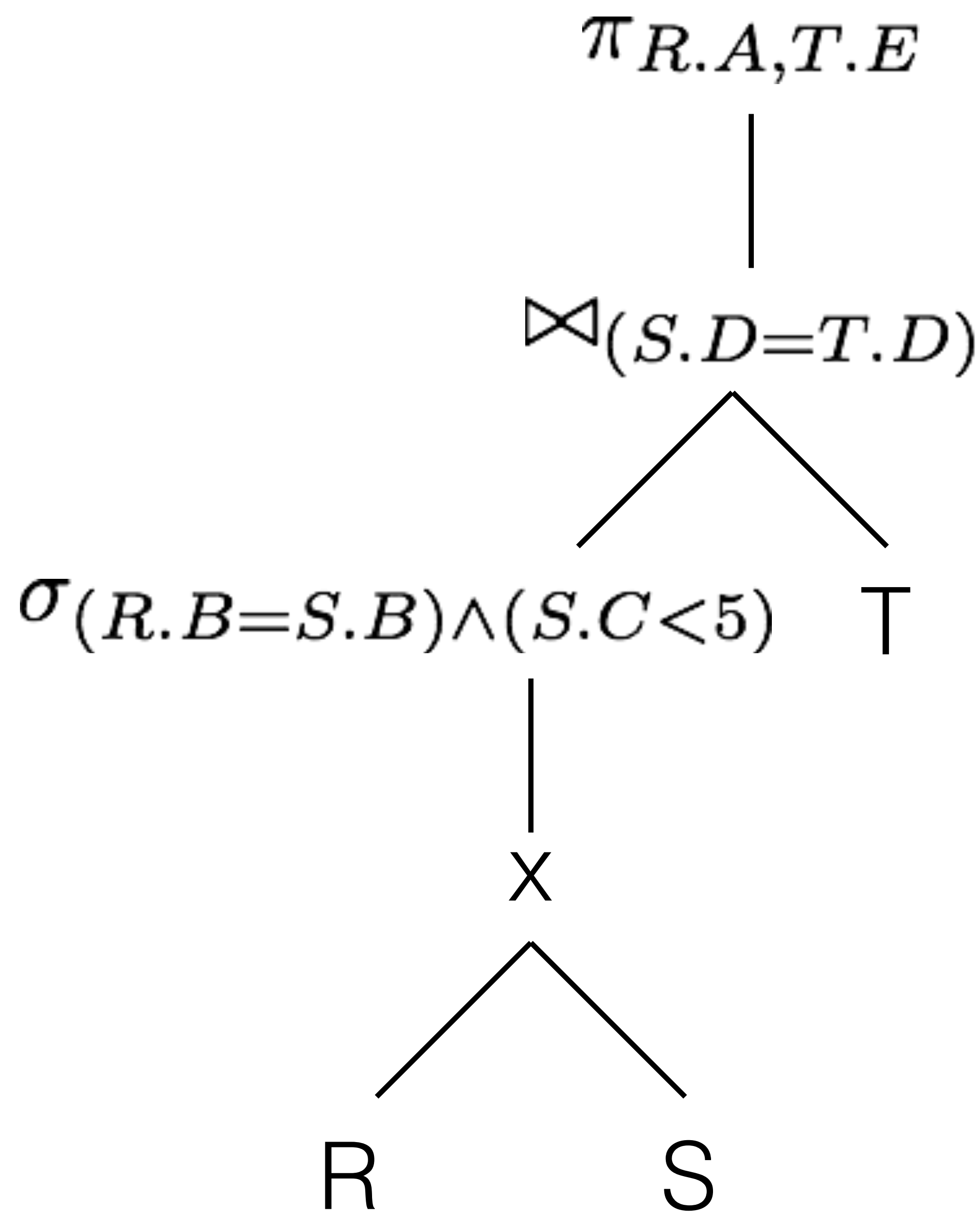
Example



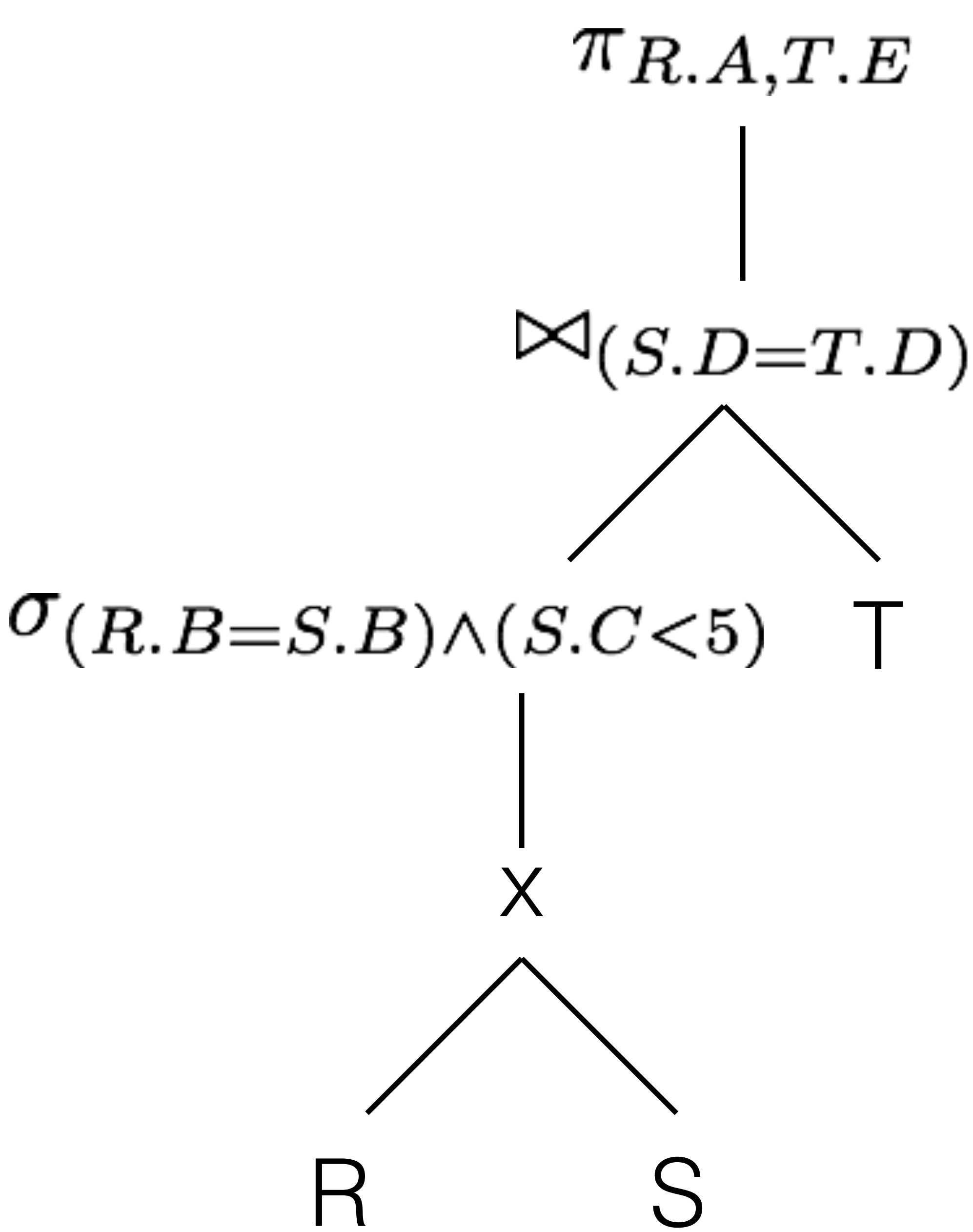
$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
X
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
X
R
S
$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
X
R
S

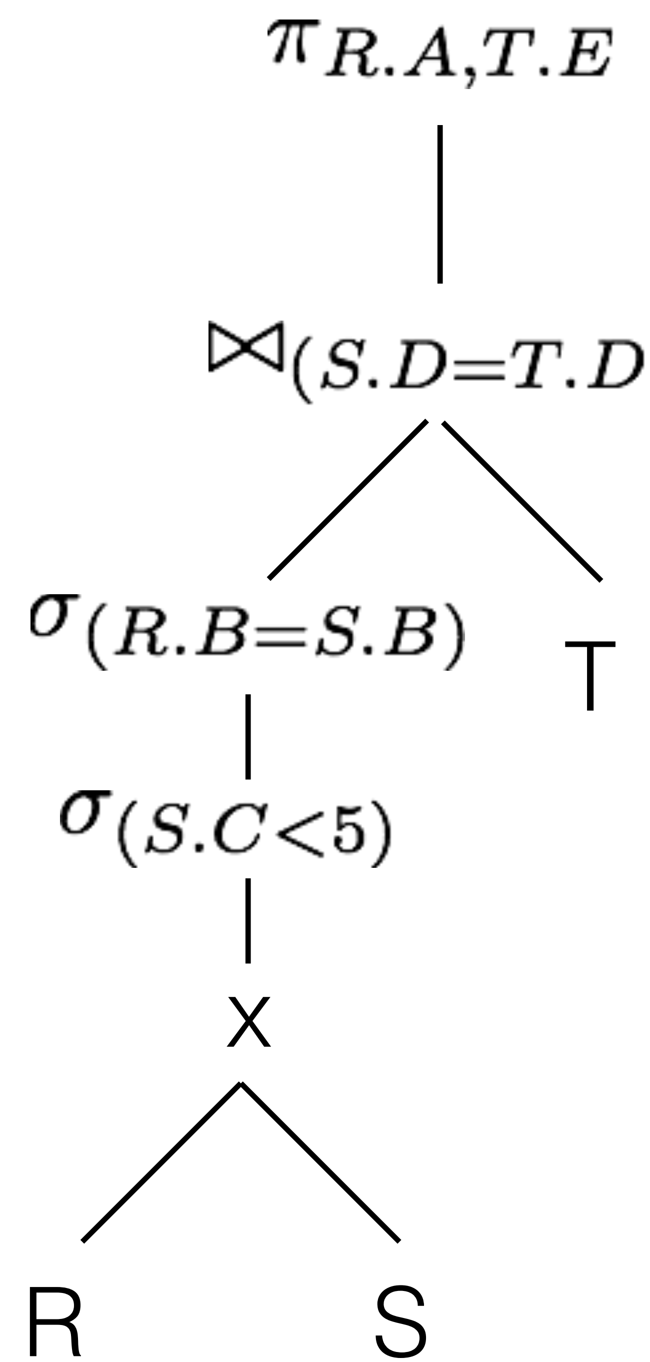
Example


$\pi_{R.A,T.E}$
$\sigma_{(S.D=T.D)}$
X
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
X
R
S

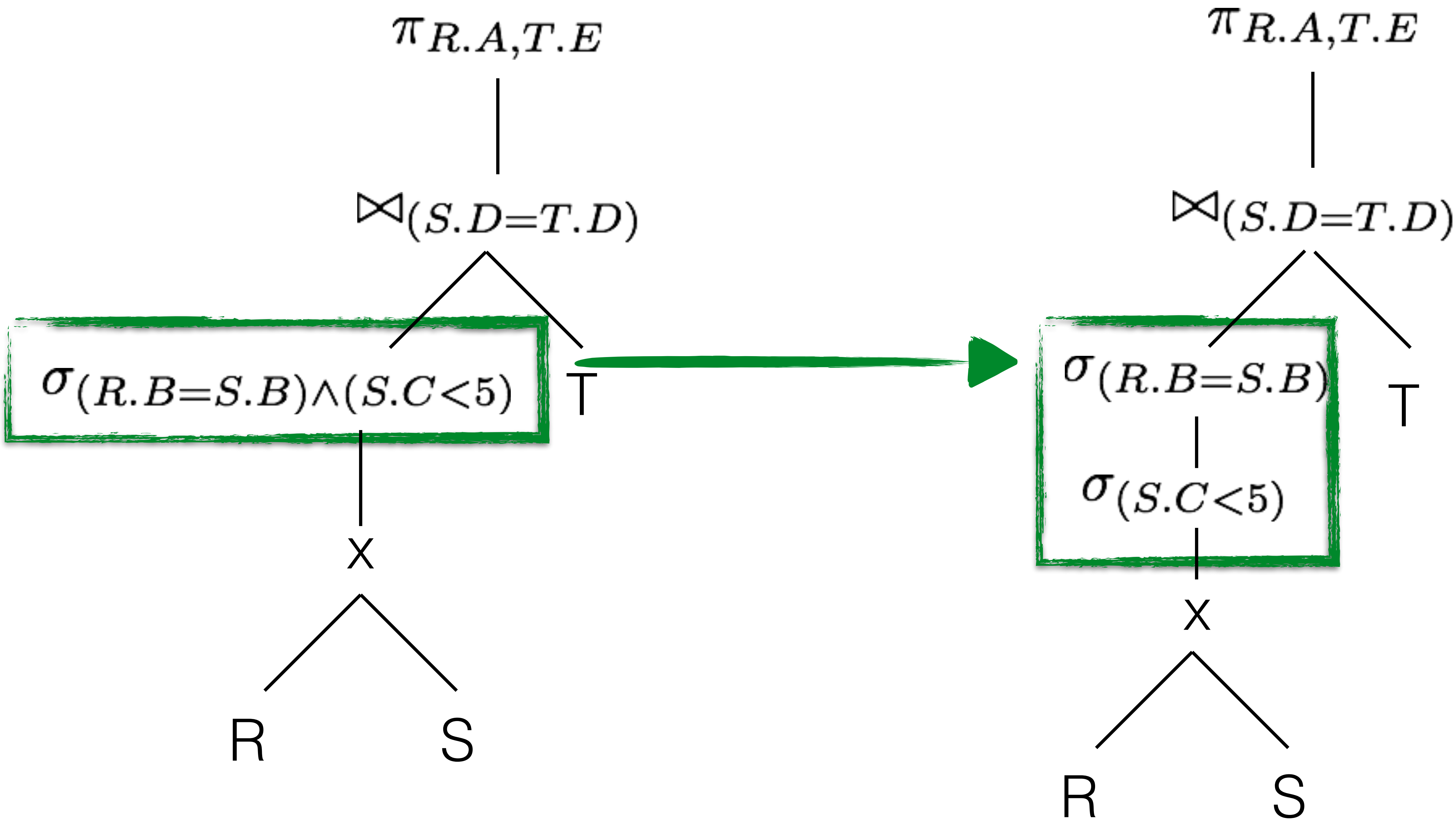

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
X
R
S

Example


$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
x
R
S

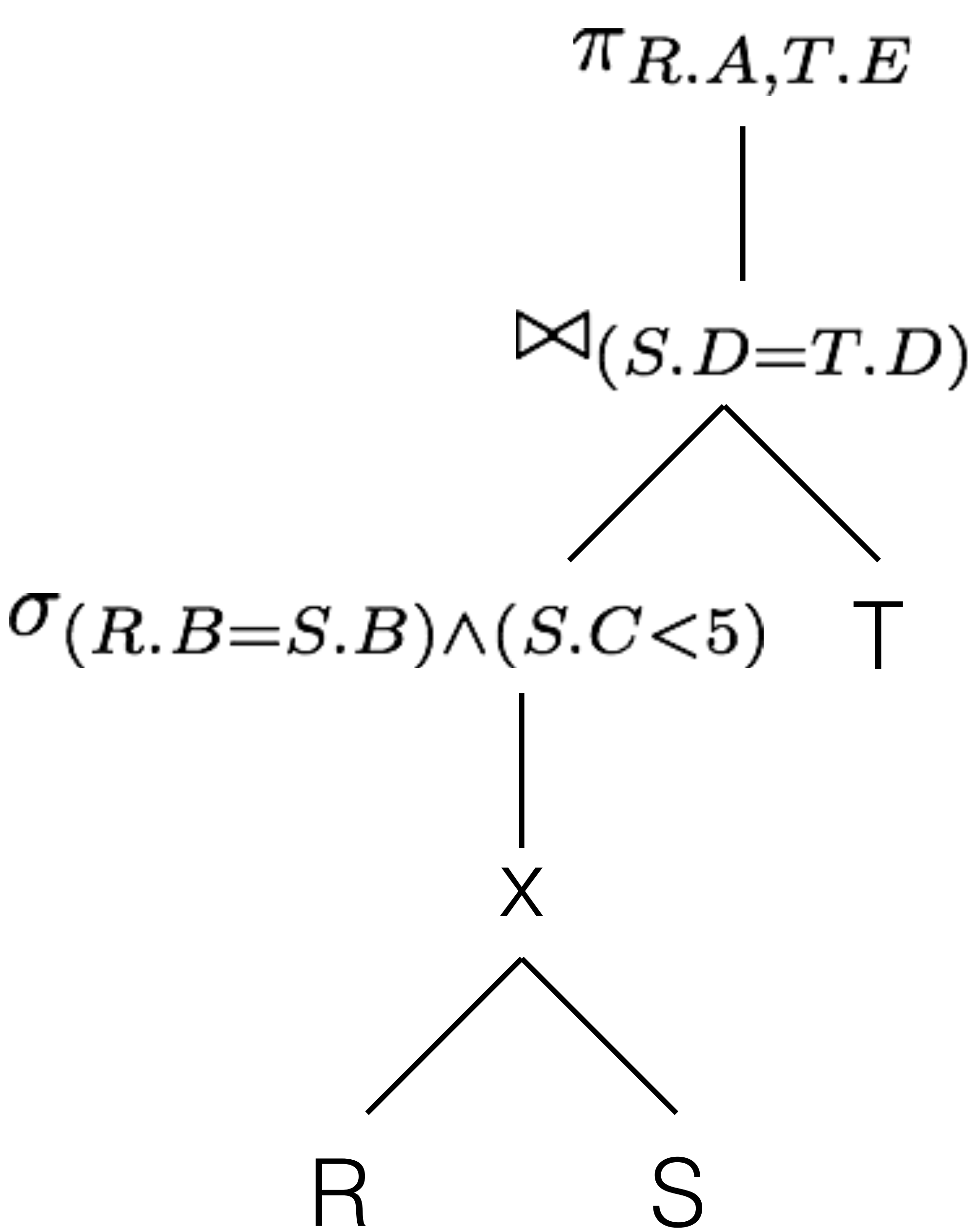

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
$\sigma_{(S.C<5)}$
x
R
S

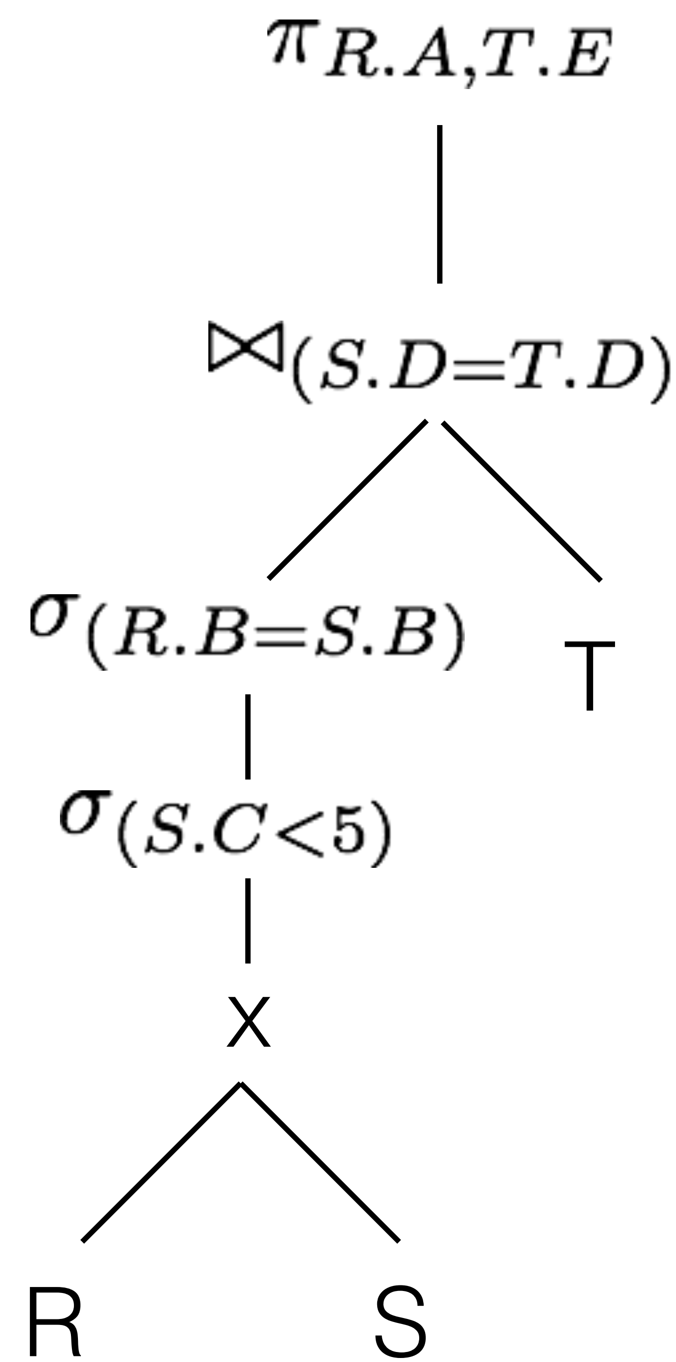
Example



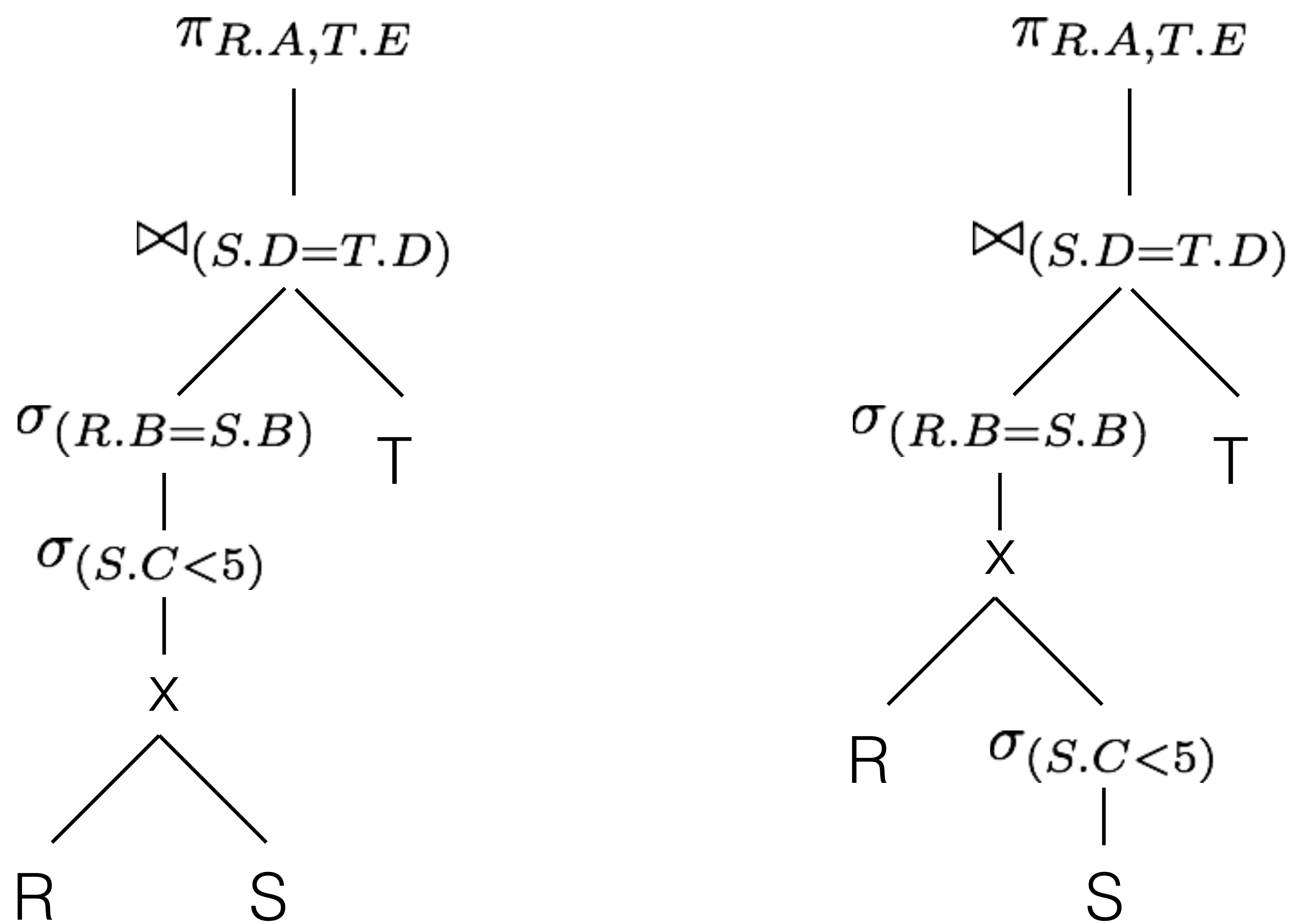
$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
X
R
S
$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
$\sigma_{(S.C<5)}$
X
R
S

Example

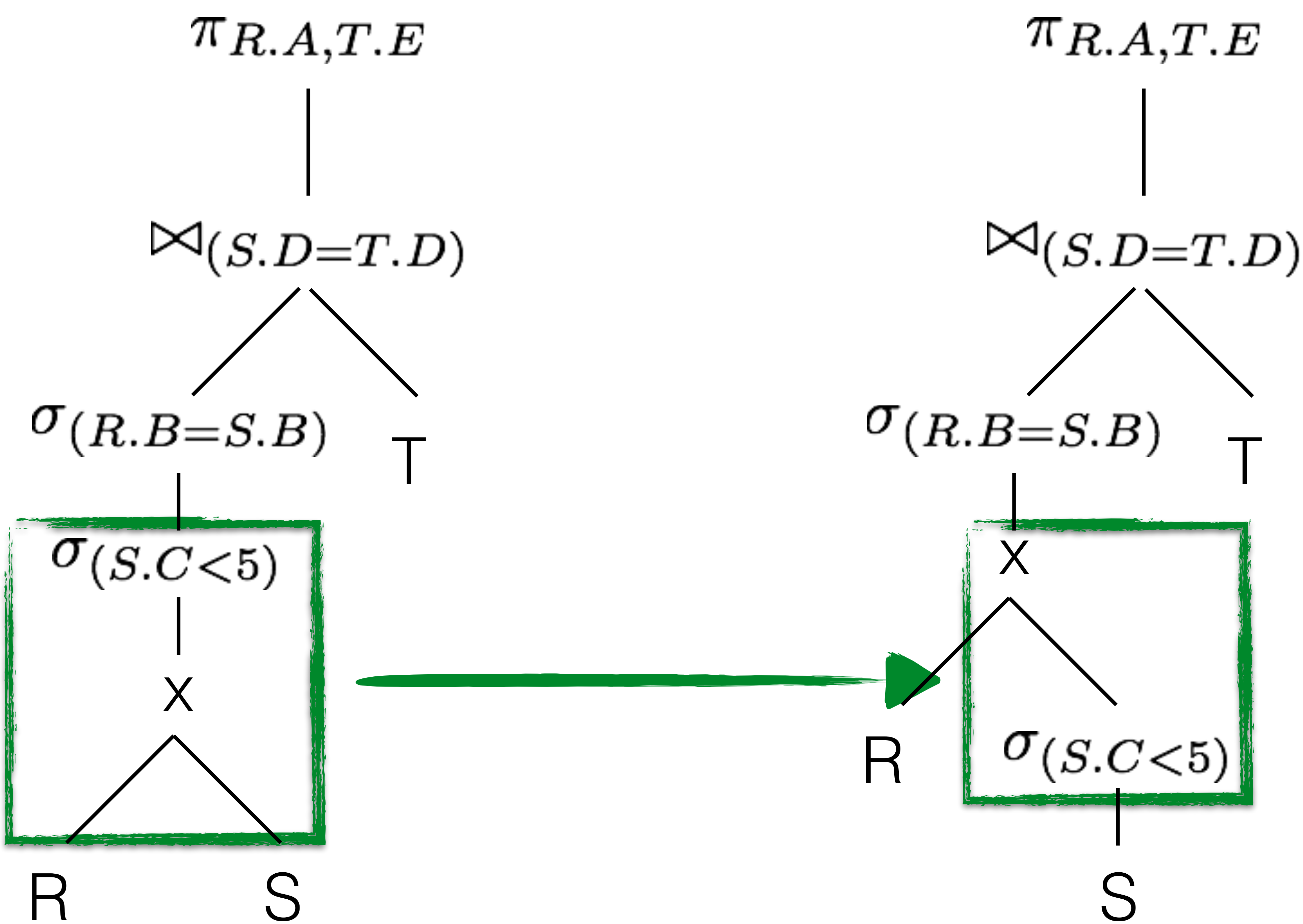

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)\wedge(S.C<5)}$
T
x
R
S


$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
$\sigma_{(S.C<5)}$
x
R
S

Example

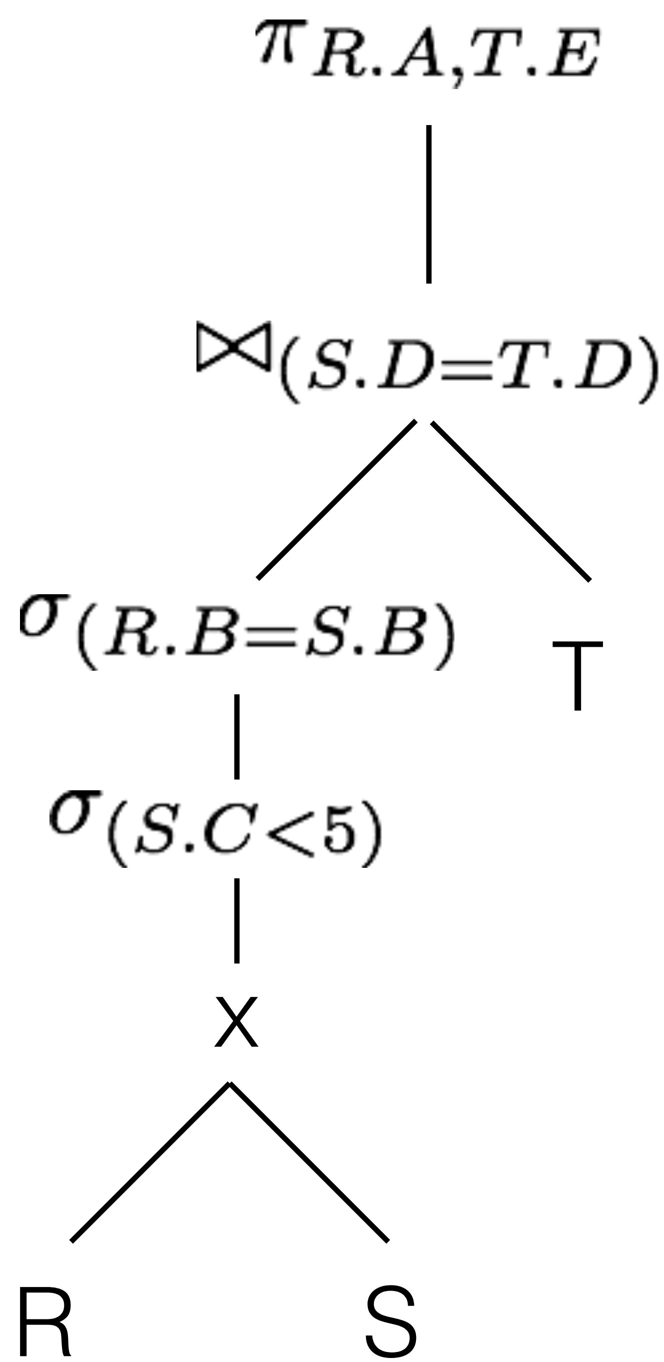

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
$\sigma_{(S.C<5)}$
X
R
S
$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
X
R
$\sigma_{(S.C<5)}$
S

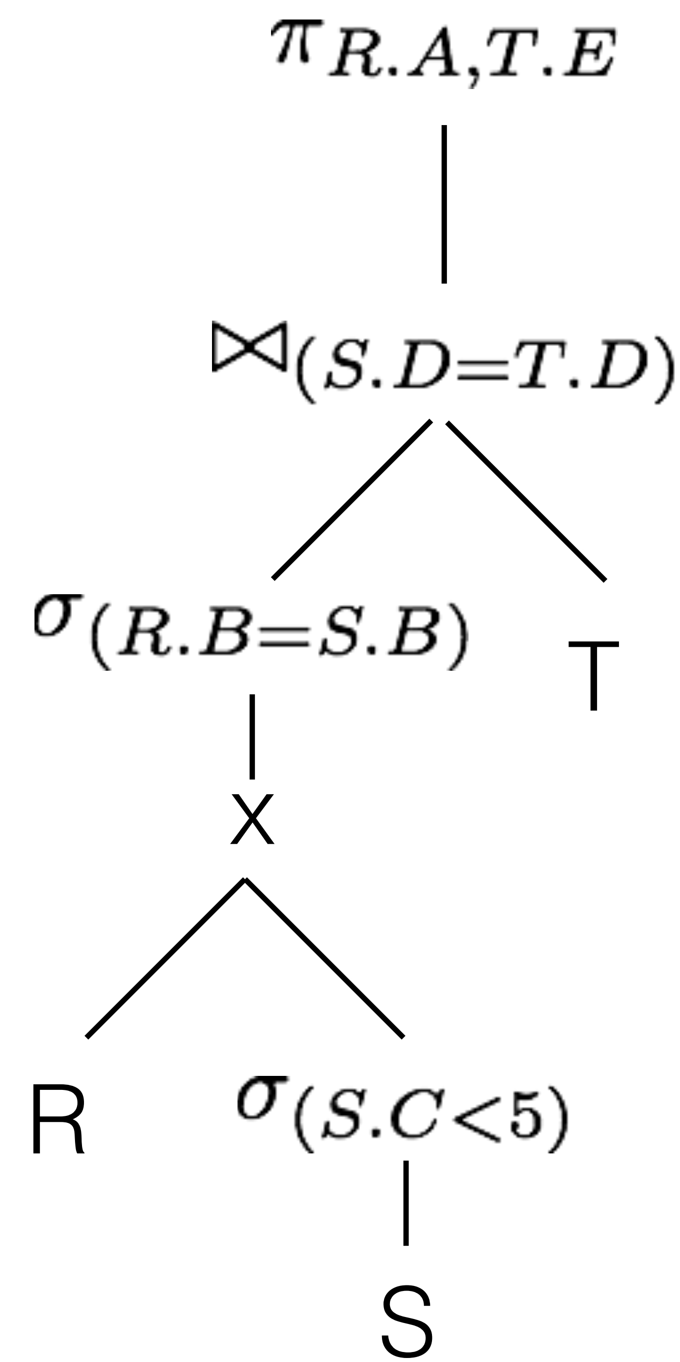
Example



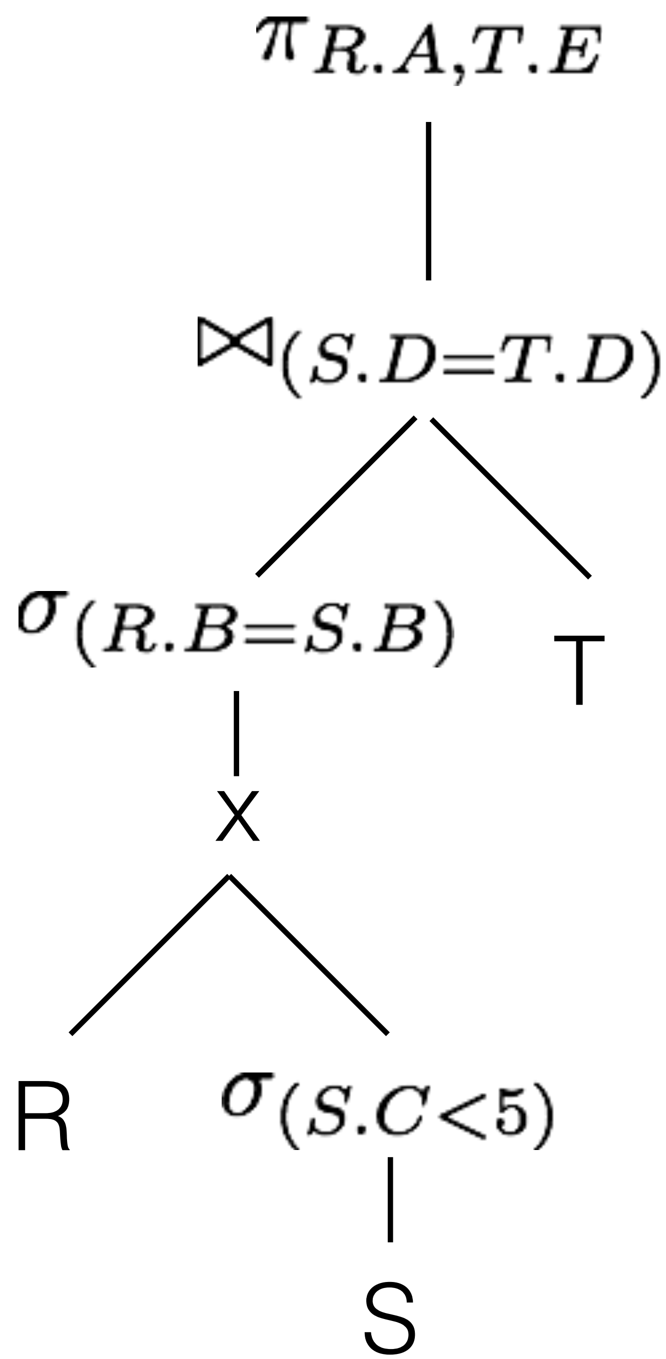
$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
$\sigma_{(S.C<5)}$
X
R
S
$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
X
R
$\sigma_{(S.C<5)}$
S

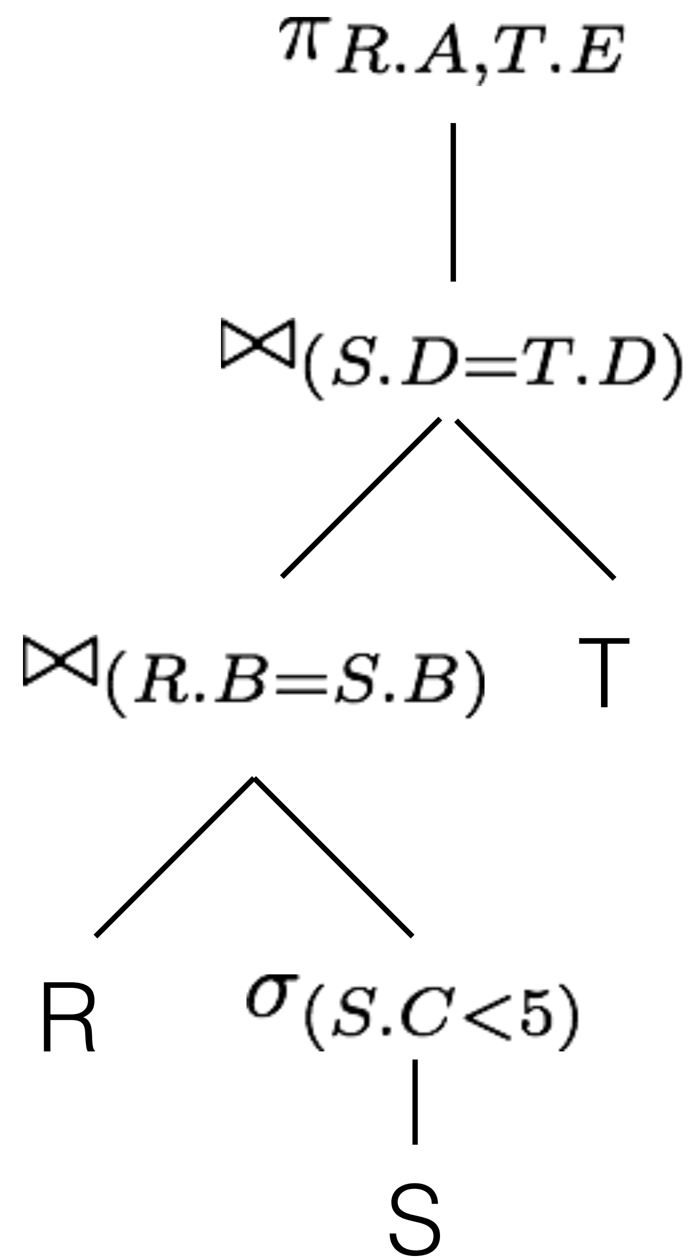
Example


$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
$\sigma_{(S.C<5)}$
X
R
S

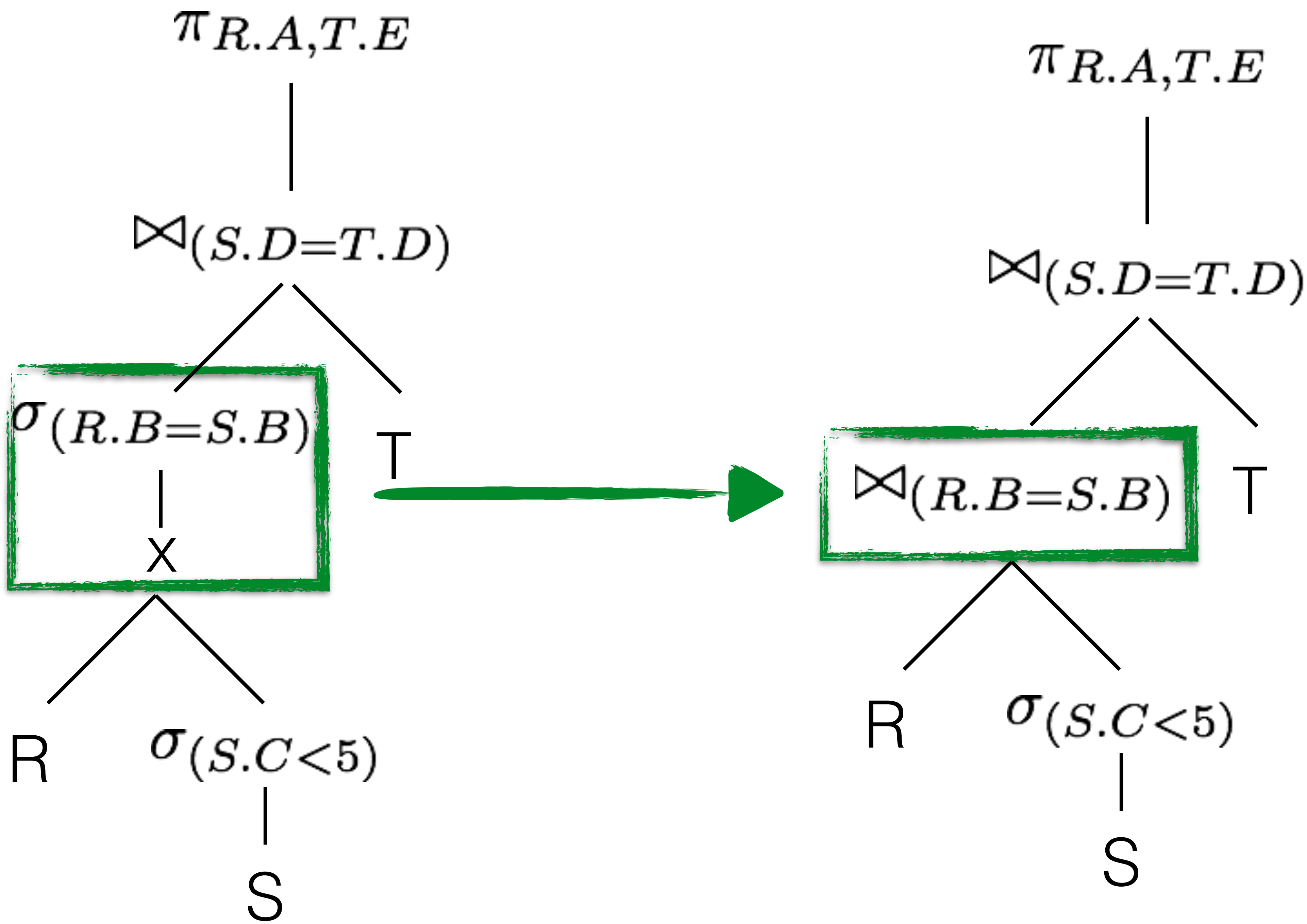

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
X
R
$\sigma_{(S.C<5)}$
S

Example

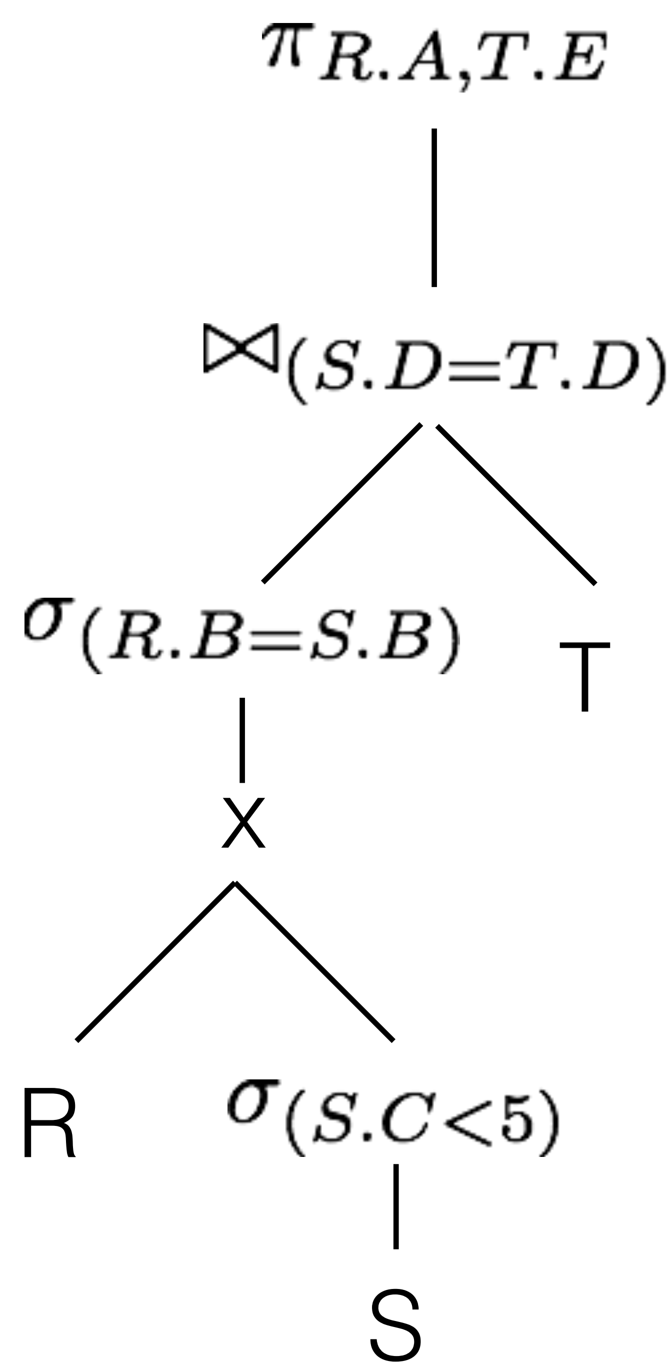

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
x
R
$\sigma_{(S.C<5)}$
S

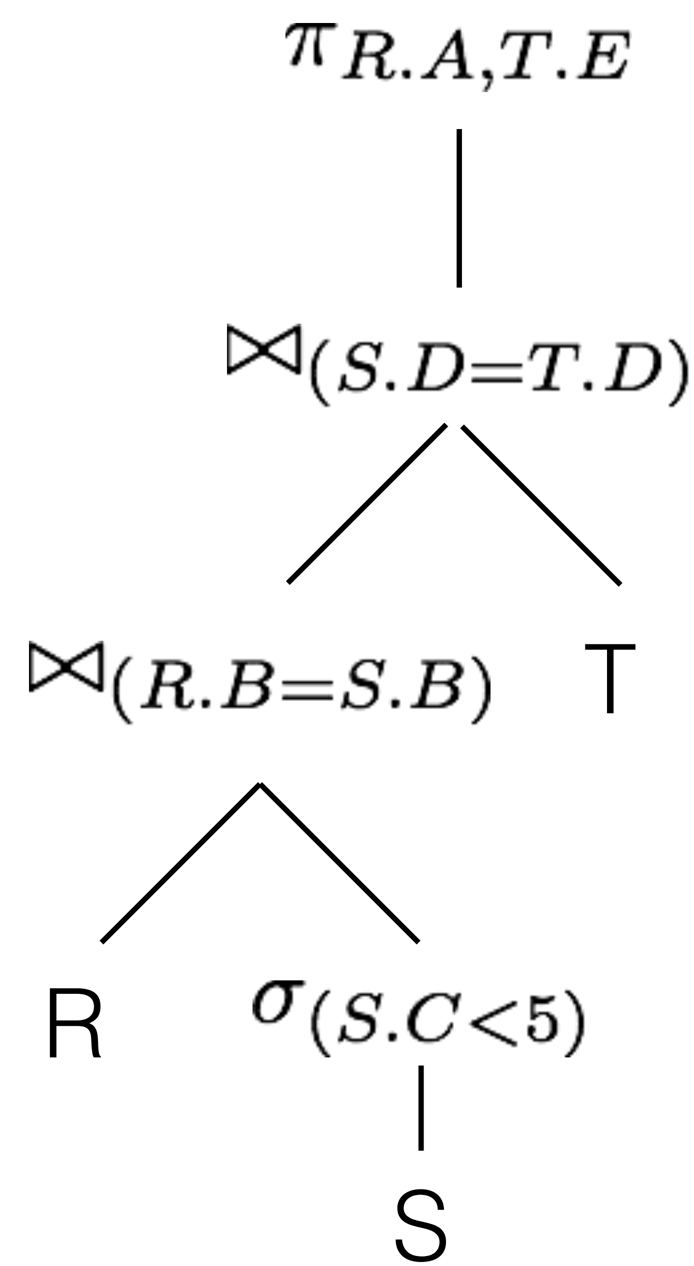

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\bowtie_{(R.B=S.B)}$
T
R
$\sigma_{(S.C<5)}$
S

Example

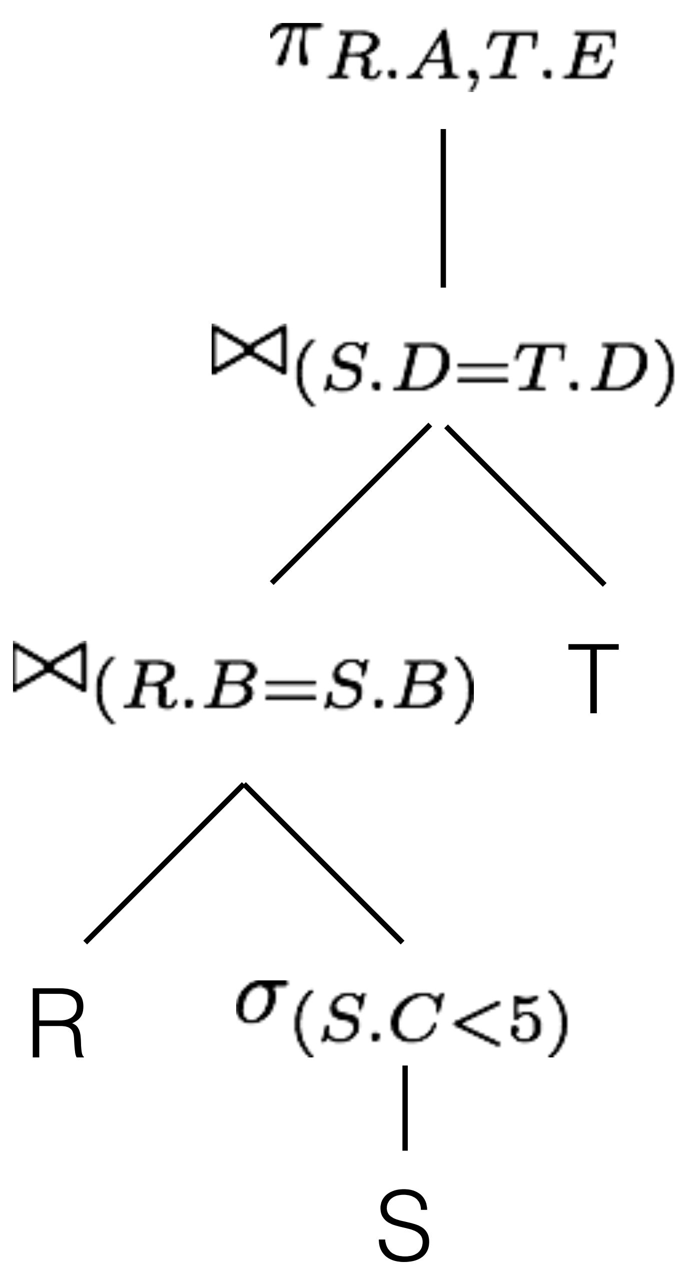

$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
X
R
$\sigma_{(S.C<5)}$
S
$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\bowtie_{(R.B=S.B)}$
T
R
$\sigma_{(S.C<5)}$
S

Example


$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\sigma_{(R.B=S.B)}$
T
X
R
$\sigma_{(S.C<5)}$
S


$\pi_{R.A,T.E}$
$\bowtie_{(S.D=T.D)}$
$\bowtie_{(R.B=S.B)}$
T
R
$\sigma_{(S.C<5)}$
S

Final Plan




```
SELECT R.A, T.E
  FROM R, S, T
 WHERE R.B = S.B
   AND S.C < 5
   AND S.D = T.D
```

Translate Dumb, Optimize Later

Find Patterns (`Select(Cross(R,S))`) …
… and Replace (`Join(R,S)`)

RA Equivalencies

$(R \bowtie S) \bowtie T$ vs $R \bowtie (S \bowtie T)$

RA Equivalencies

$$(R \bowtie S) \bowtie T \qquad \text{vs} \qquad R \bowtie (S \bowtie T)$$

Which form is better?