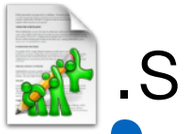
CSE4/562 Database Systems

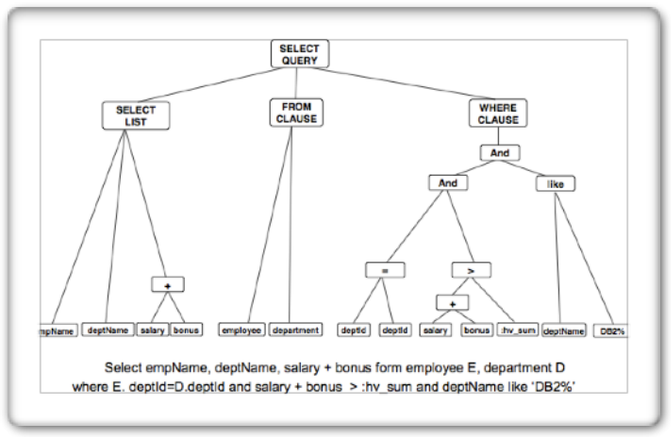
Practicum Component

02/21/2018

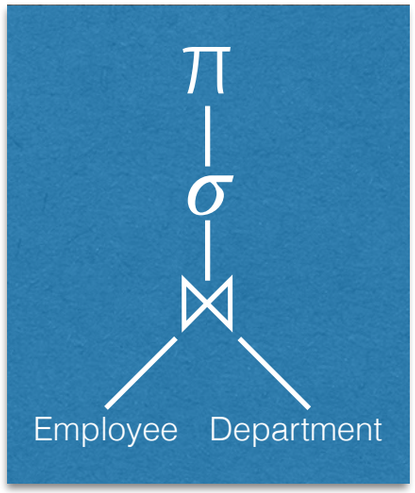
Recap



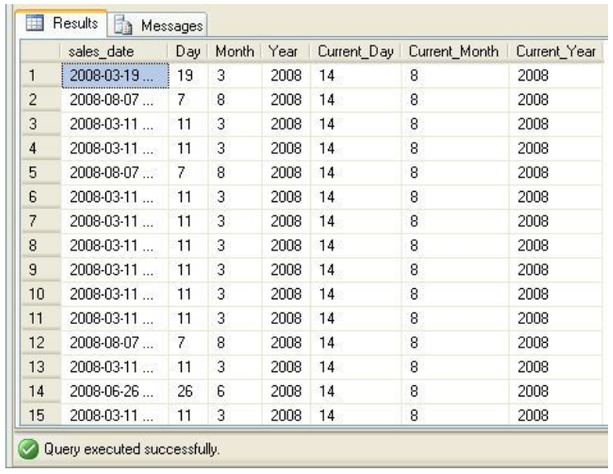
.sql



Parsed Query



Data



	sales_date	Day	Month	Year	Current_Day	Current_Month	Current_Year
1	2008-03-19 ...	19	3	2008	14	8	2008
2	2008-08-07 ...	7	8	2008	14	8	2008
3	2008-03-11 ...	11	3	2008	14	8	2008
4	2008-03-11 ...	11	3	2008	14	8	2008
5	2008-08-07 ...	7	8	2008	14	8	2008
6	2008-03-11 ...	11	3	2008	14	8	2008
7	2008-03-11 ...	11	3	2008	14	8	2008
8	2008-03-11 ...	11	3	2008	14	8	2008
9	2008-03-11 ...	11	3	2008	14	8	2008
10	2008-03-11 ...	11	3	2008	14	8	2008
11	2008-03-11 ...	11	3	2008	14	8	2008
12	2008-08-07 ...	7	8	2008	14	8	2008
13	2008-03-11 ...	11	3	2008	14	8	2008
14	2008-06-26 ...	26	6	2008	14	8	2008
15	2008-03-11 ...	11	3	2008	14	8	2008

Results Messages
Query executed successfully.

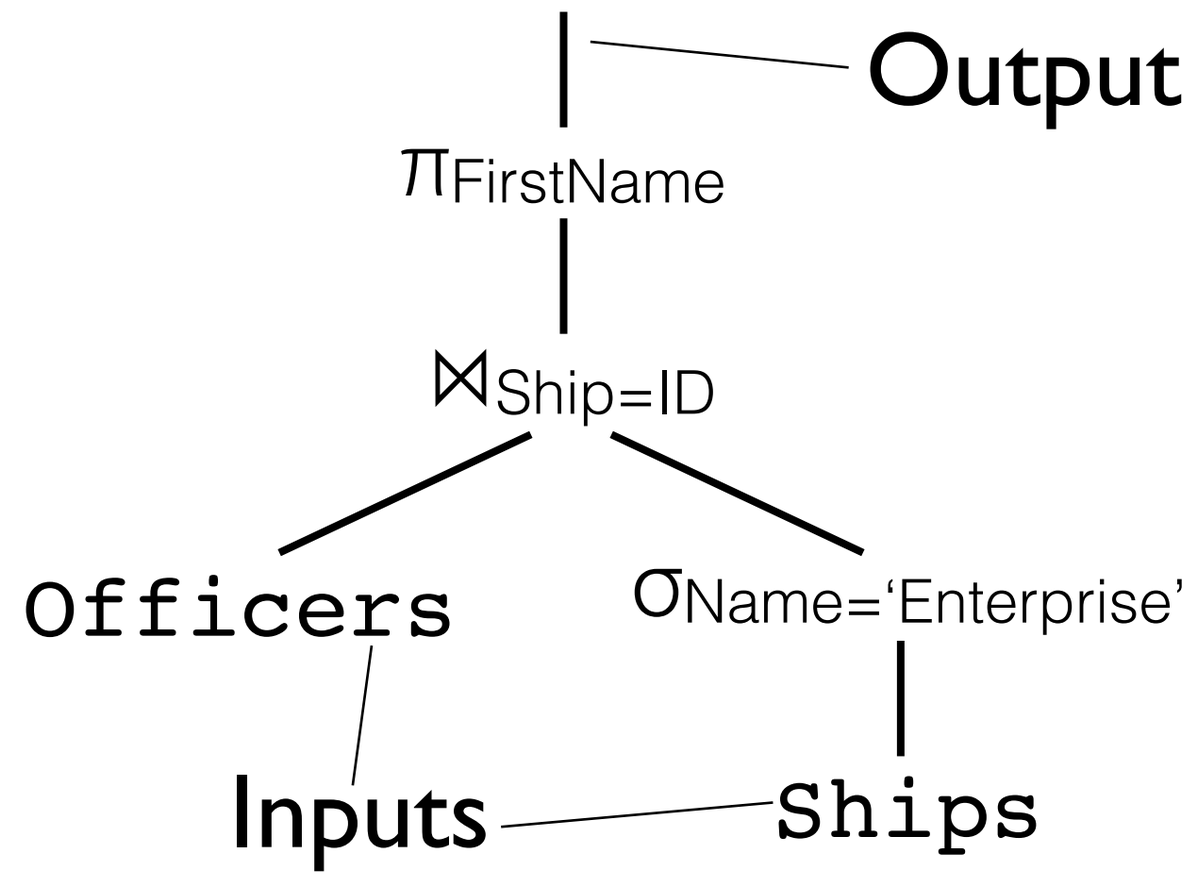
Results

Relational Algebra Trees

```
SELECT O.FirstName
FROM Officers O, Ships S
WHERE O.Ship = S.ID
      AND S.Name = 'Enterprise'
```

$\pi_{\text{FirstName}}(\texttt{Officers} \bowtie_{\text{Ship=ID}} (\sigma_{\text{Name='Enterprise'}} \texttt{Ships}))$

Relational Algebra Trees



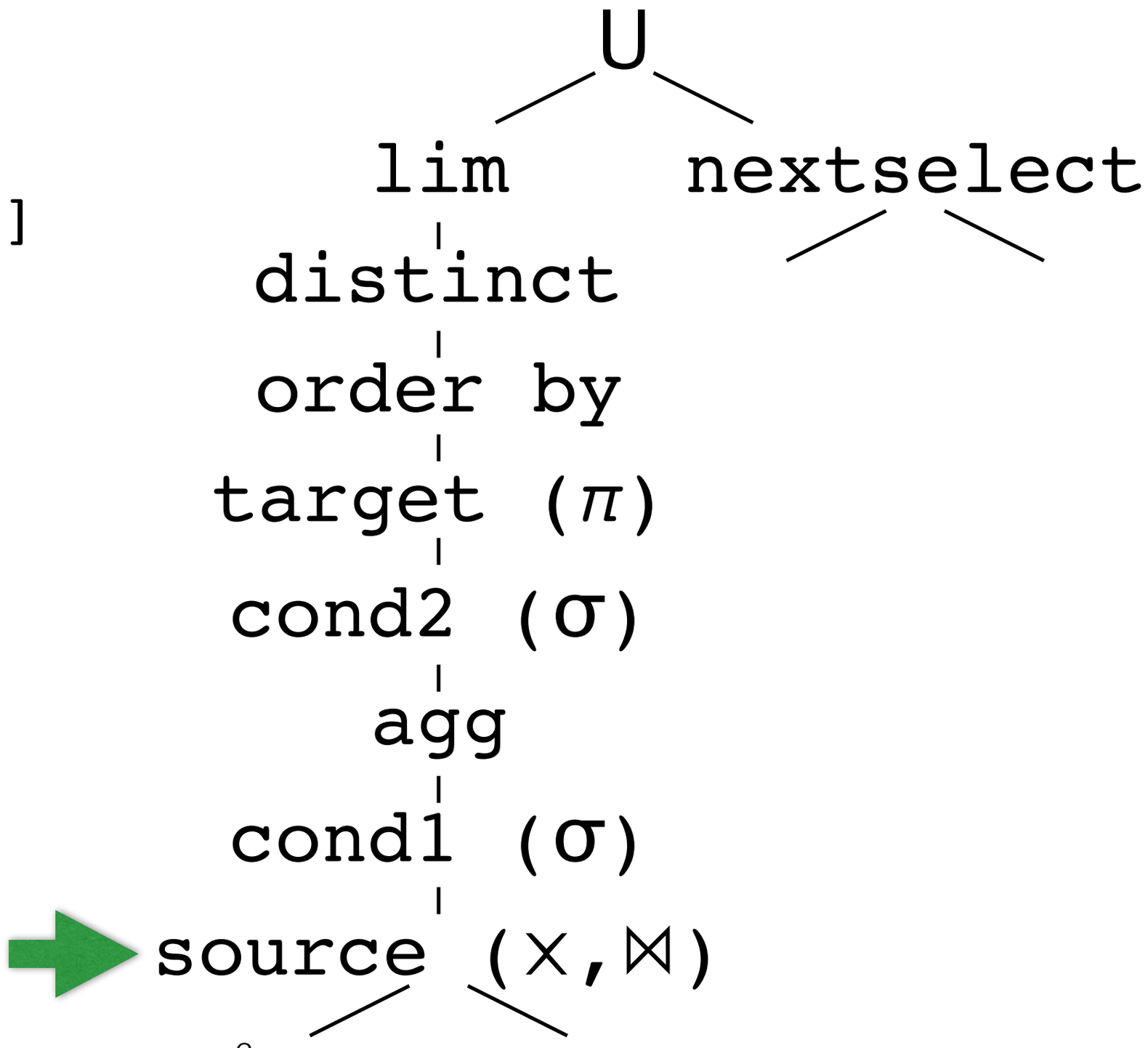
$\pi_{\text{FirstName}}(\texttt{Officers} \bowtie_{\text{Ship=ID}} (\sigma_{\text{Name='Enterprise'}} \texttt{Ships}))$

InstanceOf

```
Statement statement = parser.Statement();

if(statement instanceof Select) {
  Algebra raTree = parseTree((Select)statement);
  evaluate(raTree);

} else if(statement instanceof CreateTable) {
  loadTableSchema((CreateTable)statement);

}
```

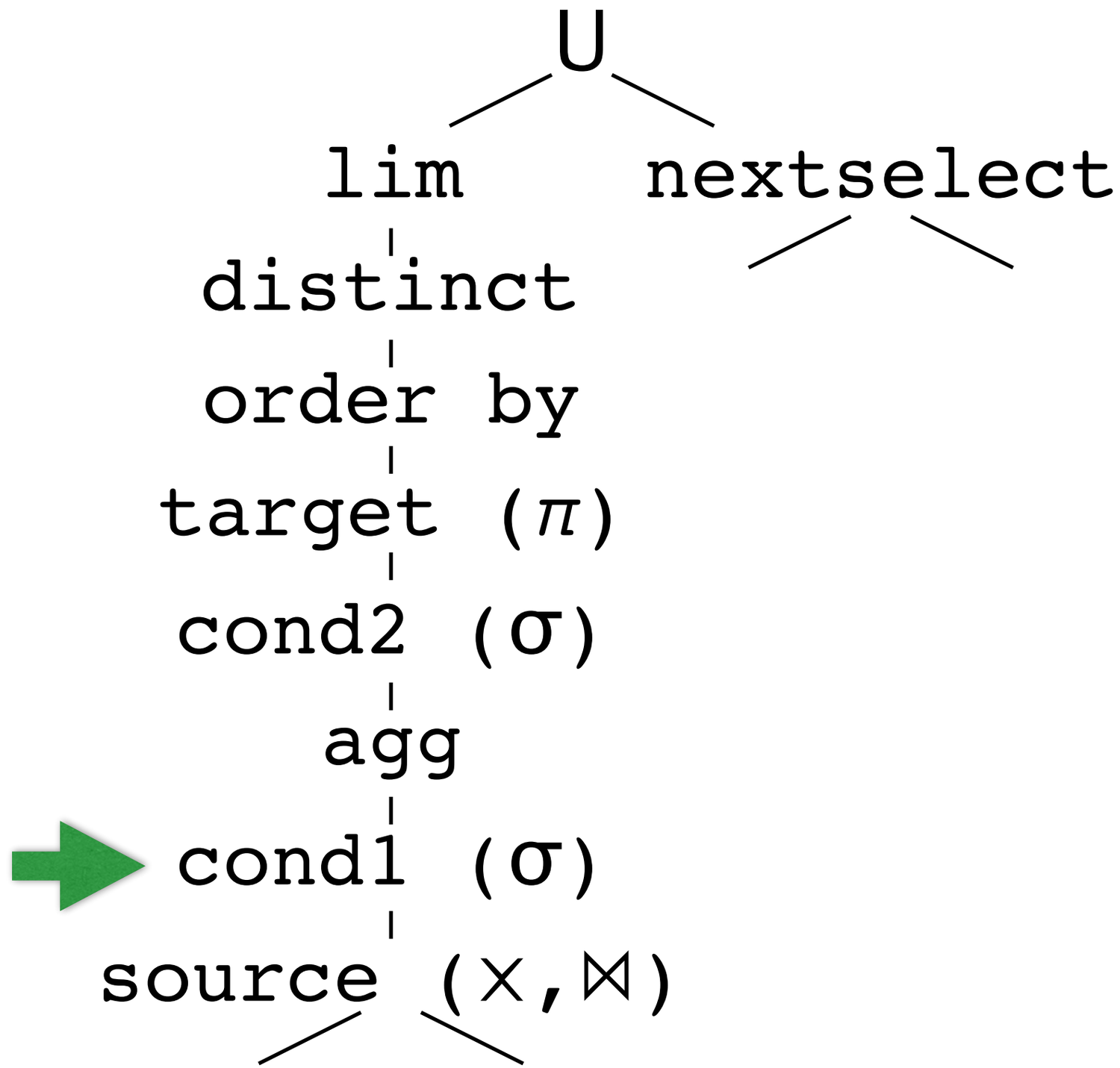
SQL to RA

```
SELECT [DISTINCT]
       target
FROM source
WHERE cond1
GROUP BY …
HAVING cond2
ORDER BY order
LIMIT lim
UNION nextselect
```

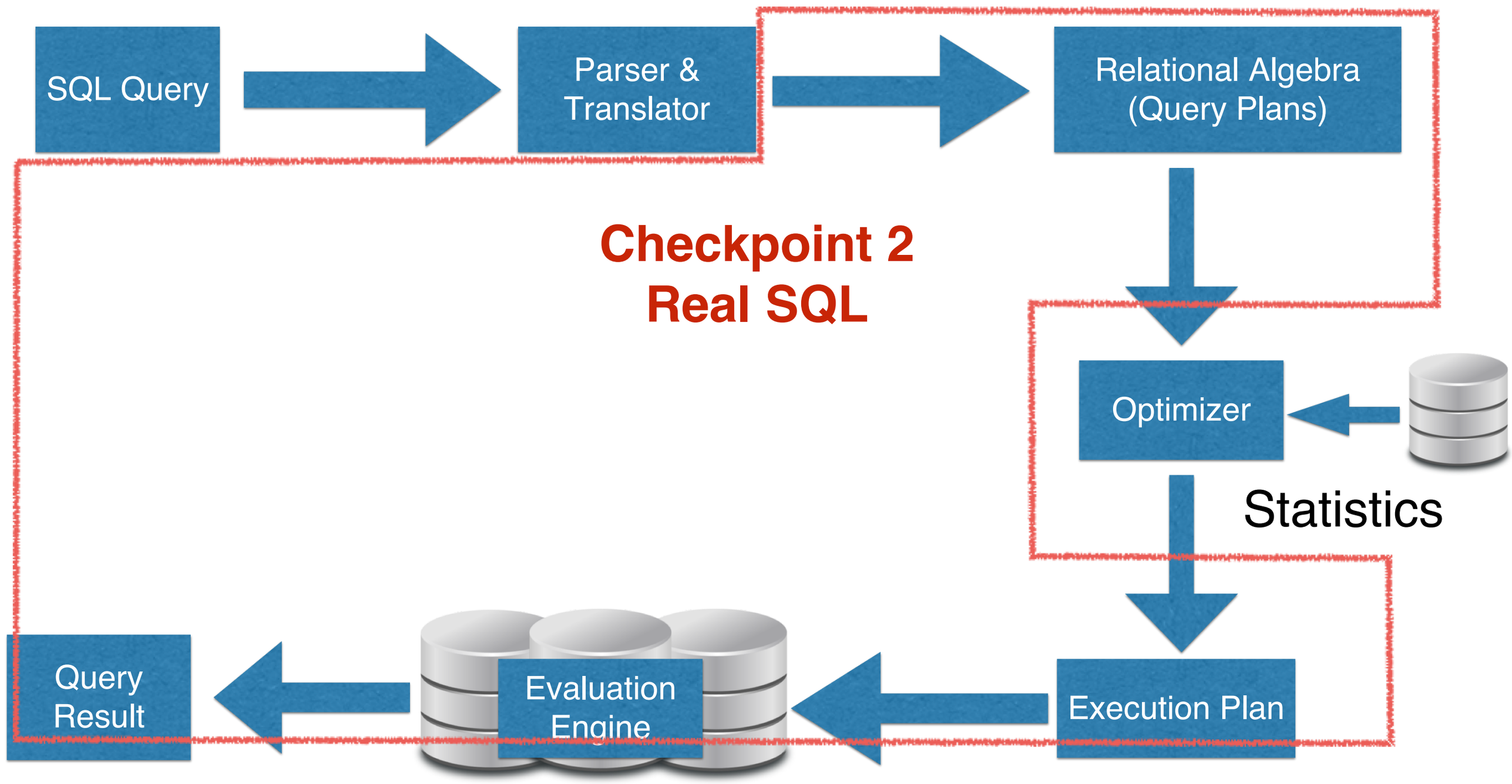


SQL to RA

```
SELECT [DISTINCT]
       target
FROM source
WHERE cond1
GROUP BY …
HAVING cond2
ORDER BY order
LIMIT lim
UNION nextselect
```

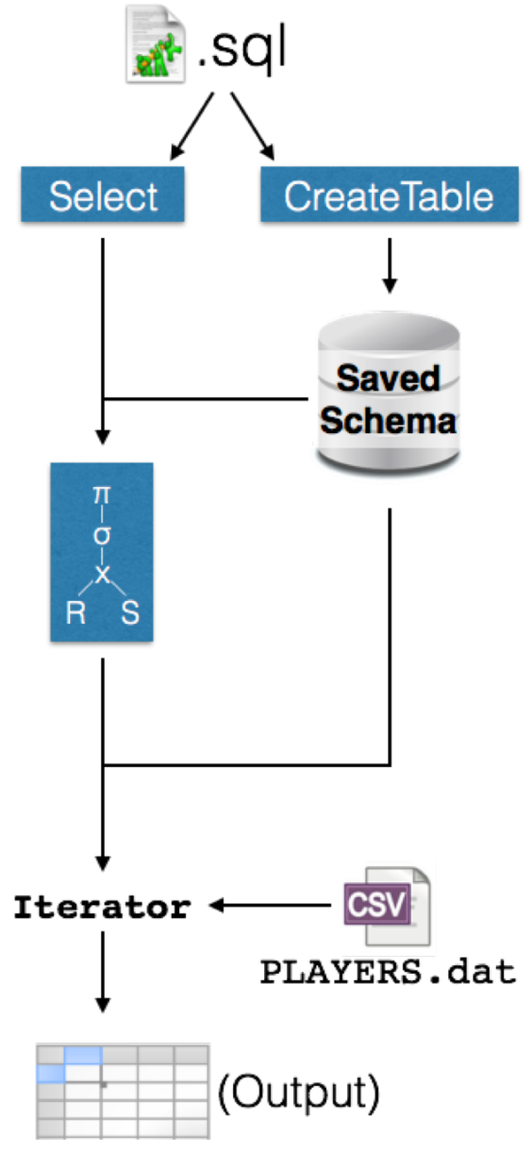


Project Outline



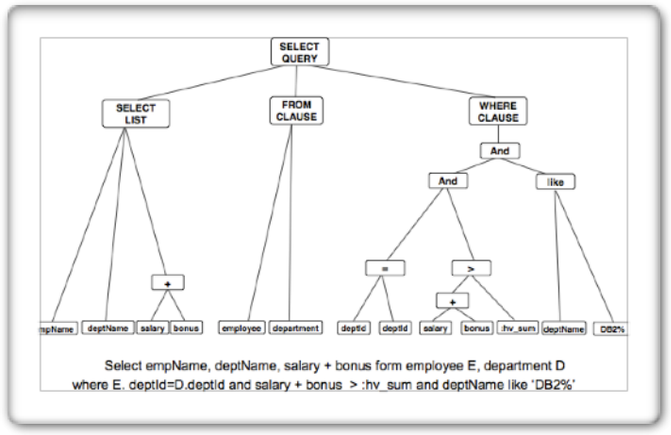
Checkpoint 2

- How do you join multiple tables, efficiently?
- How do you create a query plan?
- How do you deal with nested queries?
- Can you sort data? Just choose top-k rows?

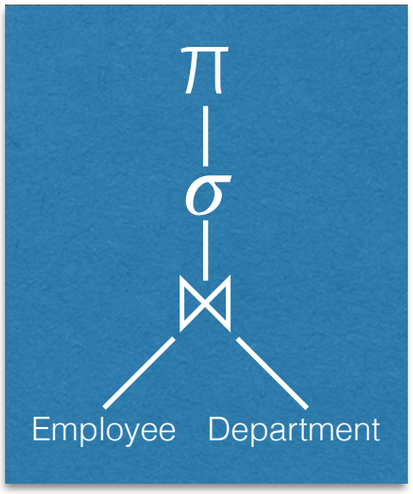


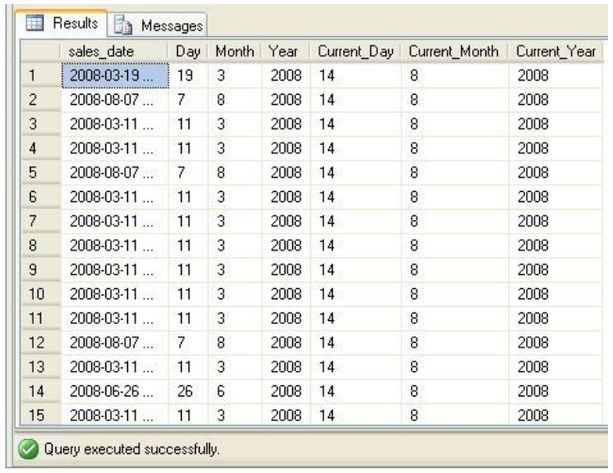
The Evaluation Pipeline

.sql



Parsed Query





	sales_date	Day	Month	Year	Current_Day	Current_Month	Current_Year
1	2008-03-19 ...	19	3	2008	14	8	2008
2	2008-08-07 ...	7	8	2008	14	8	2008
3	2008-03-11 ...	11	3	2008	14	8	2008
4	2008-03-11 ...	11	3	2008	14	8	2008
5	2008-08-07 ...	7	8	2008	14	8	2008
6	2008-03-11 ...	11	3	2008	14	8	2008
7	2008-03-11 ...	11	3	2008	14	8	2008
8	2008-03-11 ...	11	3	2008	14	8	2008
9	2008-03-11 ...	11	3	2008	14	8	2008
10	2008-03-11 ...	11	3	2008	14	8	2008
11	2008-03-11 ...	11	3	2008	14	8	2008
12	2008-08-07 ...	7	8	2008	14	8	2008
13	2008-03-11 ...	11	3	2008	14	8	2008
14	2008-06-26 ...	26	6	2008	14	8	2008
15	2008-03-11 ...	11	3	2008	14	8	2008

Results Messages
Query executed successfully.

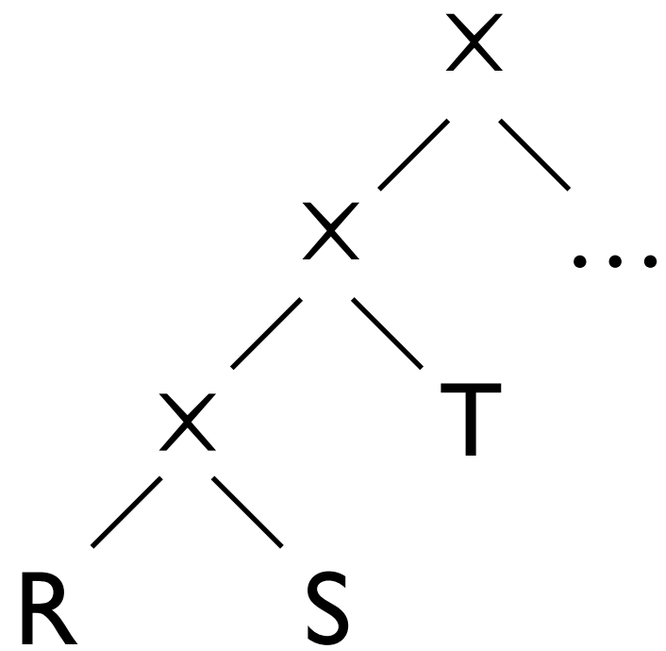
Results

Data

Checkpoint 2 focus

FROM Clause

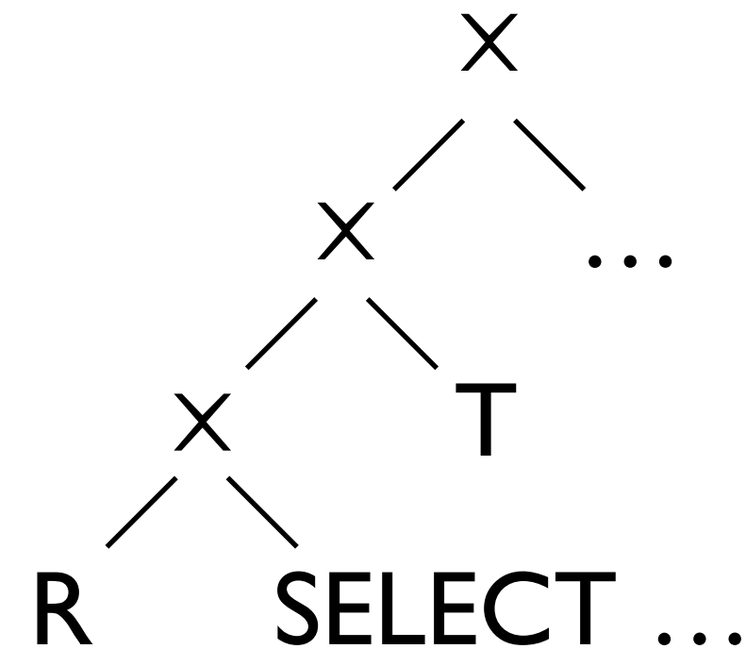
```
FROM R, S, T, …
```



What happens if I have a FROM-nested query?

FROM Clause

```
FROM R, (SELECT …) S, T, …
```



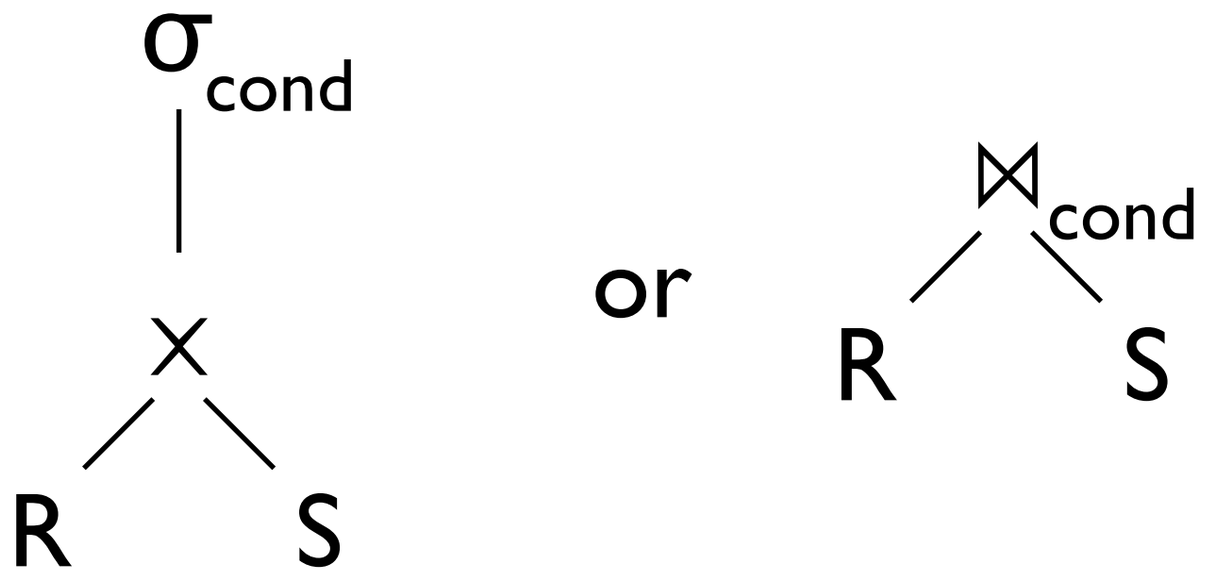
Selects are just relations!

FROM Clause

```
FROM R JOIN S ON cond
```

FROM Clause

```
FROM R JOIN S ON cond
```

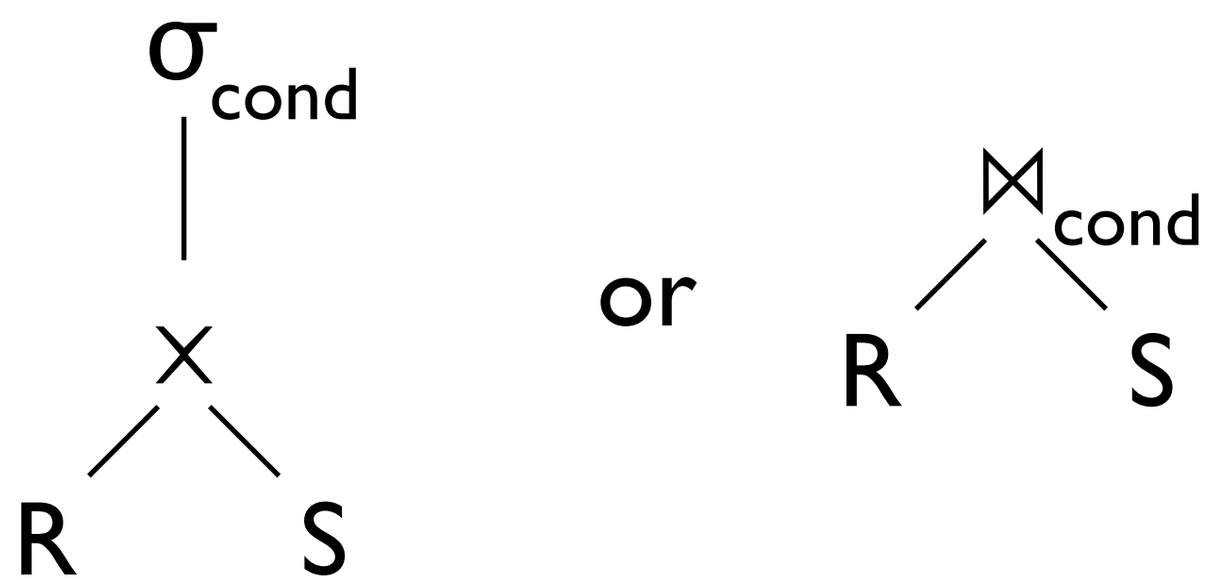


FROM Clause

```
FROM R NATURAL JOIN S
```

FROM Clause

```
FROM R NATURAL JOIN S
```

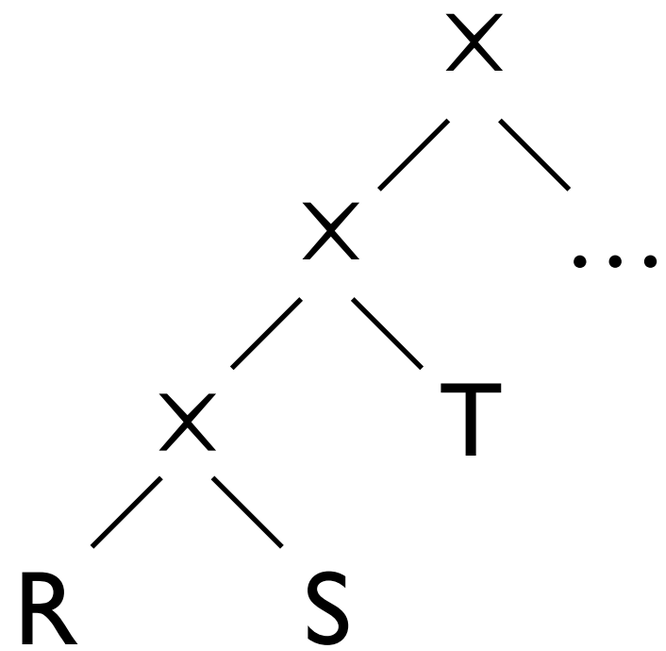



```
cond = schema(R) ∩ schema(S)
```

You need to be able to compute the schema of a RA operator

WHERE Clause - Join

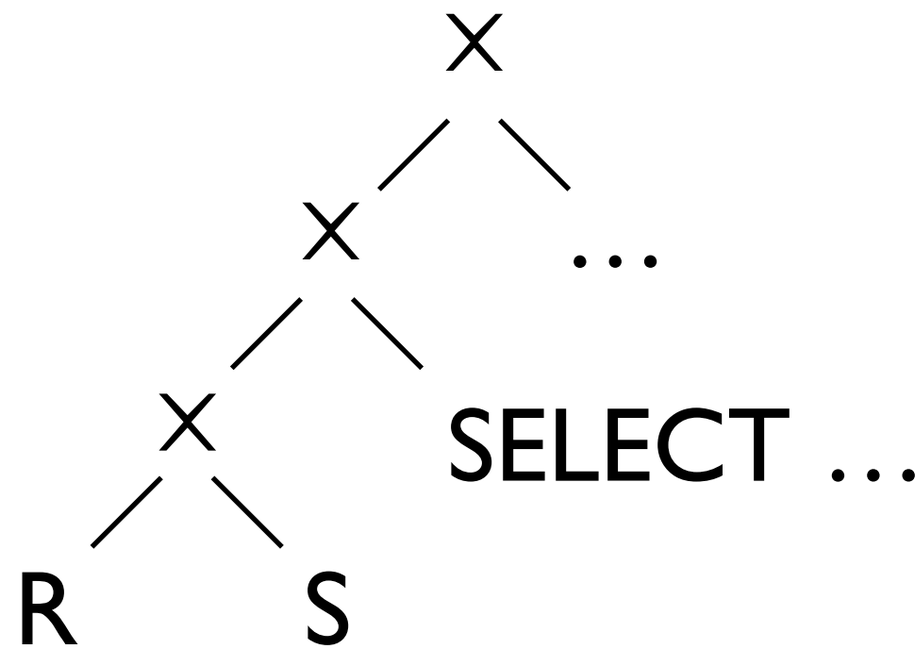
```
WHERE R.a = S.b AND S.c = T.d
```



What happens if I have a nested query in where clause?

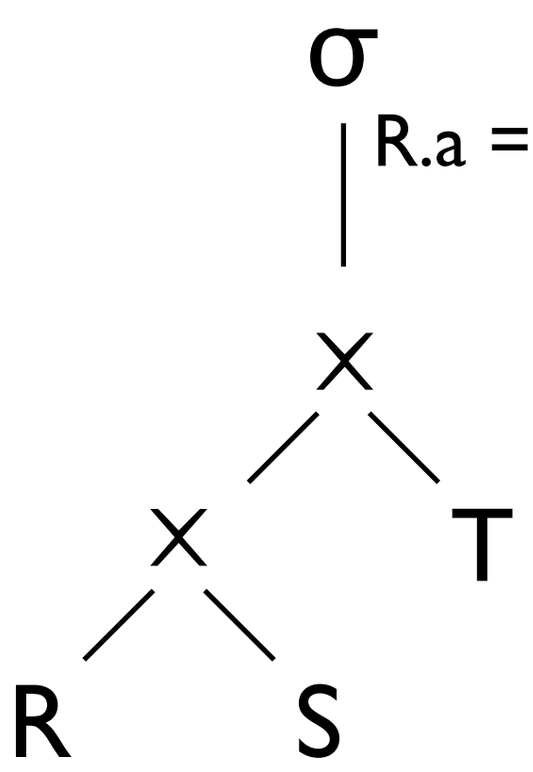
WHERE Clause - Join

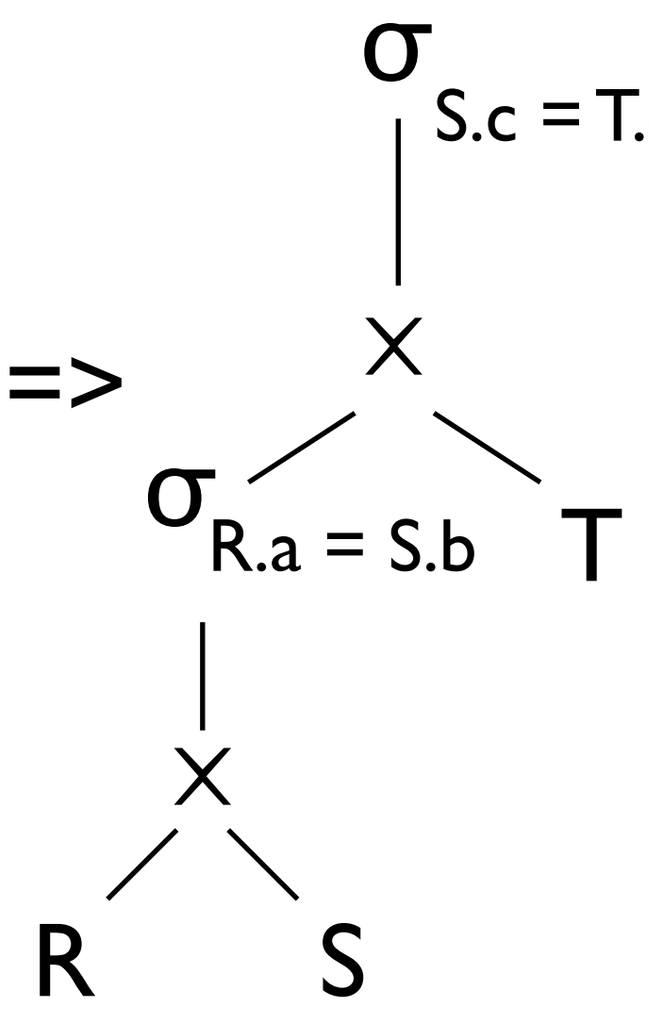
```
WHERE R.a = S.b AND S.c = (SELECT ..)
```



Creating Joins

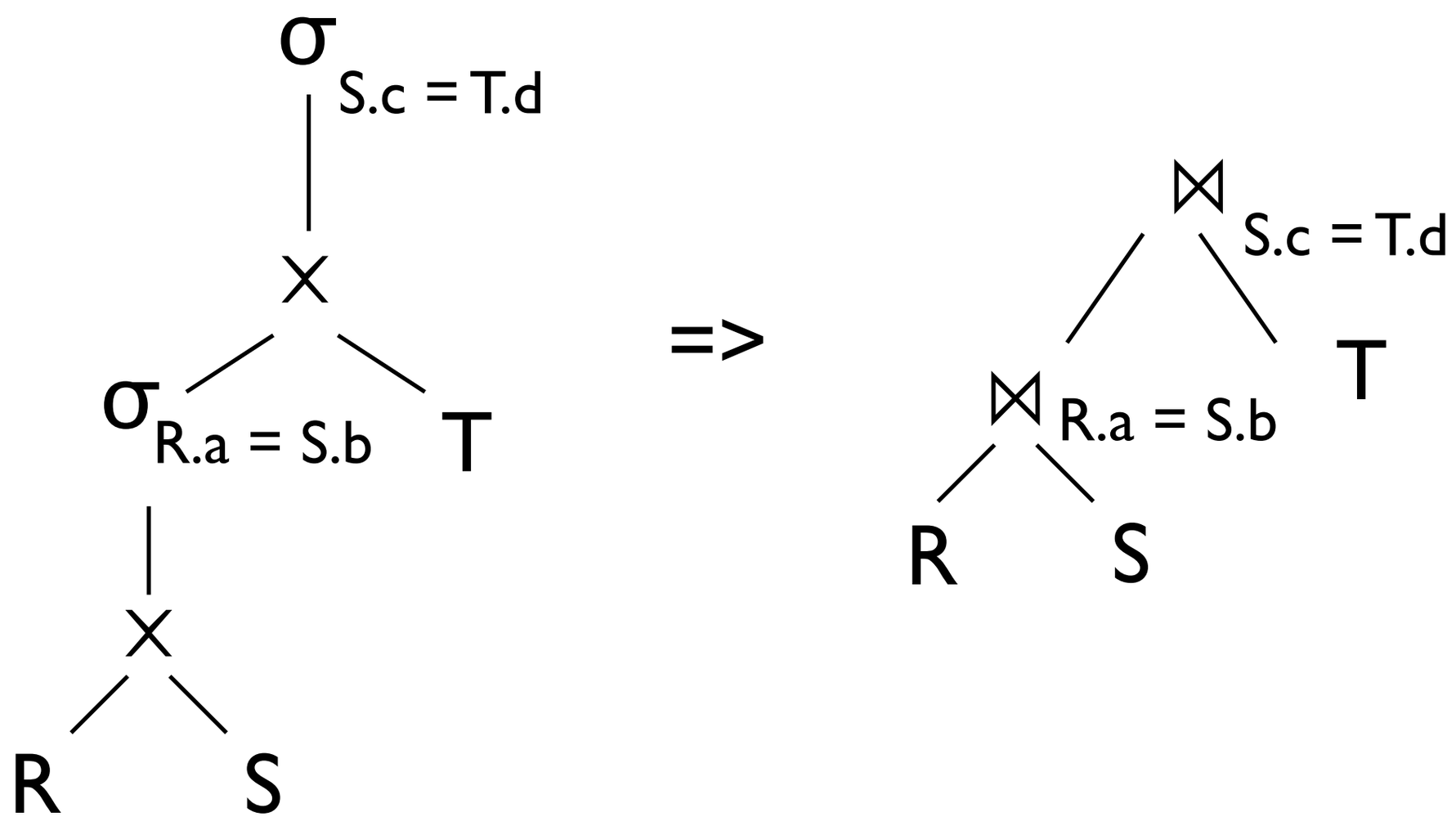
```
FROM R, S, T WHERE R.a = S.b AND S.c = T.d
```





Creating Joins

```
FROM R, S, T WHERE R.a = S.b AND S.c = T.d
```



ORDER BY

Ascending or Descending

```
SELECT Name, GamesPlayed
FROM Players
ORDER BY GamesPlayed
```

```
SELECT Name, GamesPlayed
FROM Players
ORDER BY GamesPlayed DESC
```

GetNext()

Order By

Read Each Tuple From Child

↓

Collections.sort()

(for now)

↓

Return Tuple

one by one

LIMIT

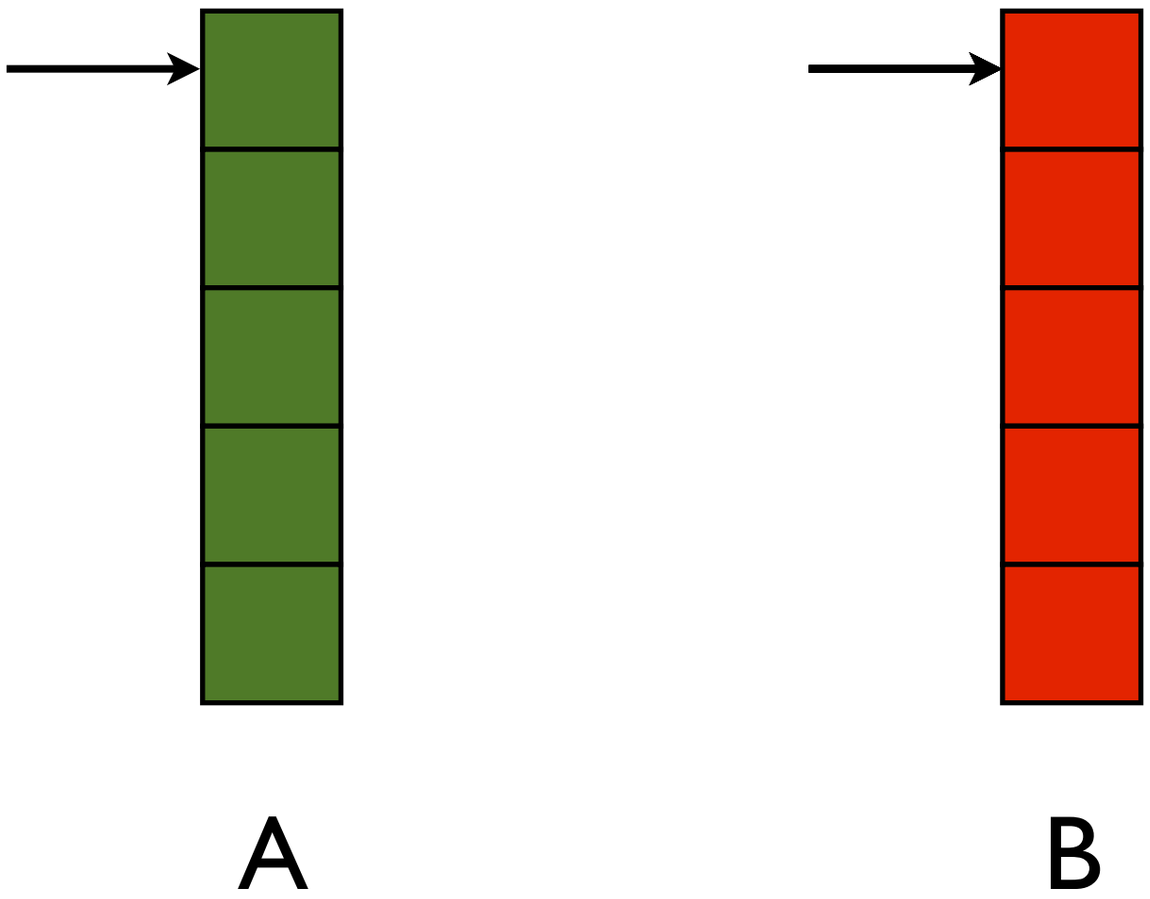
Ascending or Descending

```
SELECT Name, GamesPlayed
FROM Players
ORDER BY GamesPlayed
LIMIT 5
```

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 2) NLJ on each pair of blocks