Query Cost Estimation

CS 4750
Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.15]
[C.M. Ricardo, S.D. Urban, “Databases Illuminated, Ch.13]

Review 1: SQL and RA

Consider the following schema statements.

```
student(ID, name, dept_name, tot_cred)
takes(ID, course_ID, sec_id, semester, year, grade)
```

Find IDs and names of all students who have taken more than 3 courses

1. Write SQL query
2. Draw an RA plan

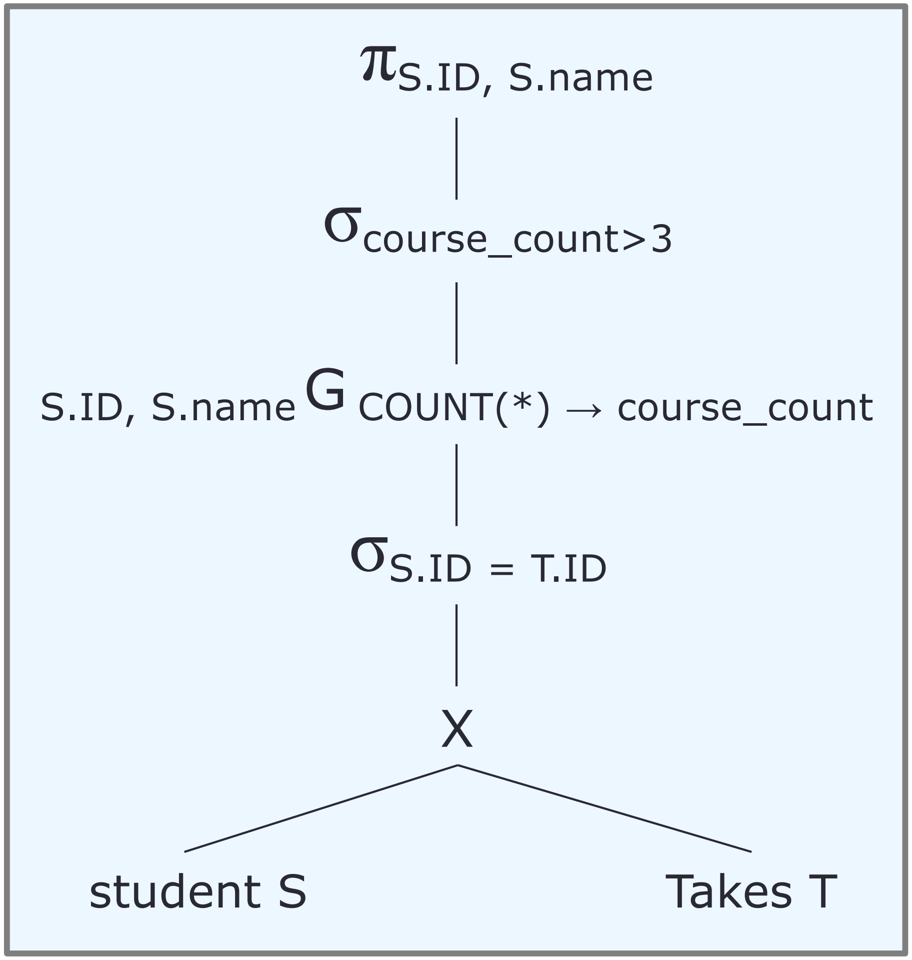
Review 1: SQL and RA (solution)

```
student(ID, name, dept_name, tot_cred)
takes(ID, course_ID, sec_id, semester, year, grade)
```

Find IDs and names of all students who have taken more than 3 courses

```
SELECT DISTINCT S.ID, S.name
FROM student S, takes T
WHERE S.ID = T.ID
GROUP BY S.ID, S.name
HAVING COUNT(*) > 3
```

$$\pi_{S.ID,\ S.\ name}\left(\sigma_{course\_count\ >3}\left({}_{S.ID,\ S.name}G_{COUNT(*)\ \rightarrow\ course\_count}\left(\sigma_{S.ID\ =\ T.ID}\left(\rho_{(S)}\,student\ \times\ \rho_{(T)}\,takes\right)\right)\right)\right)$$



$\pi_{S.ID,\ S.name}$
$\sigma_{course\_count>3}$
$_{S.ID,\ S.name}G_{COUNT(*)\ \rightarrow\ course\_count}$
$\sigma_{S.ID\ =\ T.ID}$
X
student S
Takes T

Review 2: SQL and RA

Consider the following schema statements.

```
emp(empno, ename, job, mgr, hiredate, salary, comm, deptno)
dept(deptno, dname, loc)
```

Find the names of departments where more than three employees are working

1. Write SQL query
2. Draw an RA plan

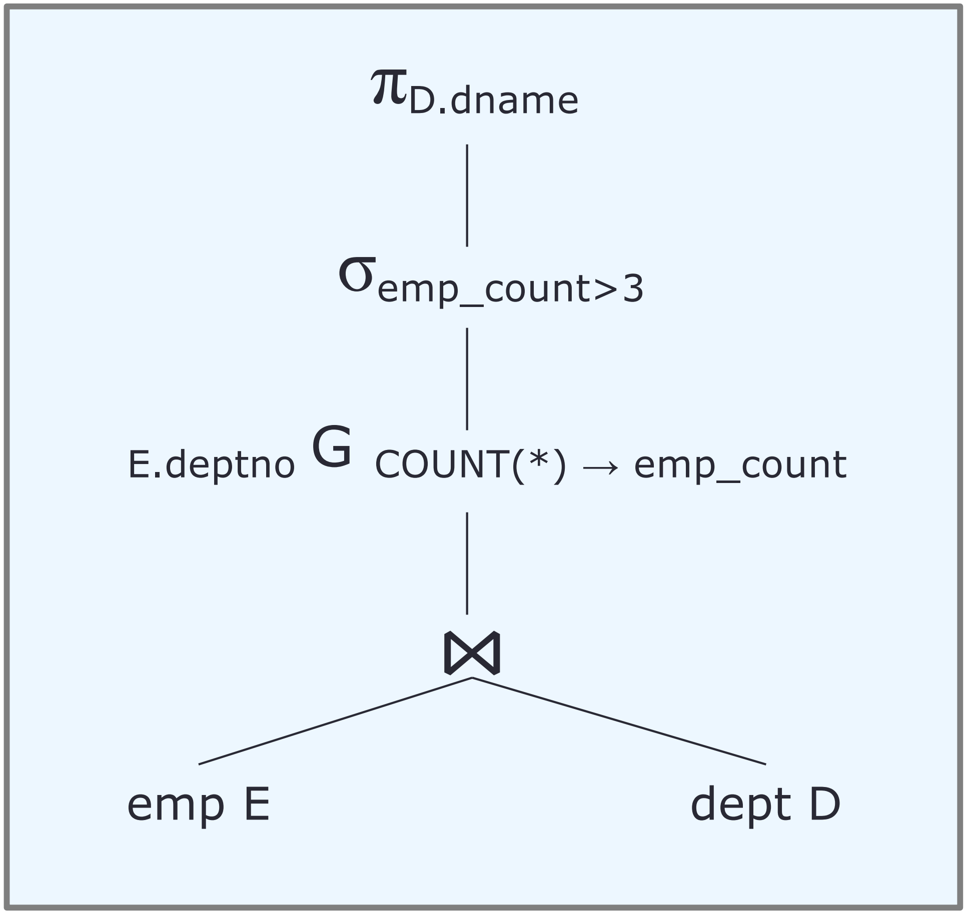
Review 2: SQL and RA (solution 1)

emp(<u>empno</u>, ename, job, mgr, hiredate, salary, comm, deptno)
dept(<u>deptno</u>, dname, loc)

Find the names of departments where more than three employees are working

```
SELECT D.dname
FROM emp E
    NATURAL JOIN dept D
GROUP BY E.deptno
HAVING COUNT(*)>3;
```

$$\pi_{D.dname}(\sigma_{emp\_count>3}({}_{E.deptno}G_{COUNT(*) \rightarrow emp\_count}(\rho_{(E)}emp \bowtie \rho_{(D)}dept)))$$



$\pi_{D.dname}$

$\sigma_{emp\_count>3}$

E.deptno G COUNT(*) → emp_count

$\bowtie$

emp E

dept D

Review 2: SQL and RA (solution 2)

```
emp(empno, ename, job, mgr, hiredate, salary, comm, deptno)
dept(deptno, dname, loc)
```

Find the names of departments where more than three employees are working

Can you think of another solution?

Review 2: SQL and RA (solution 2)

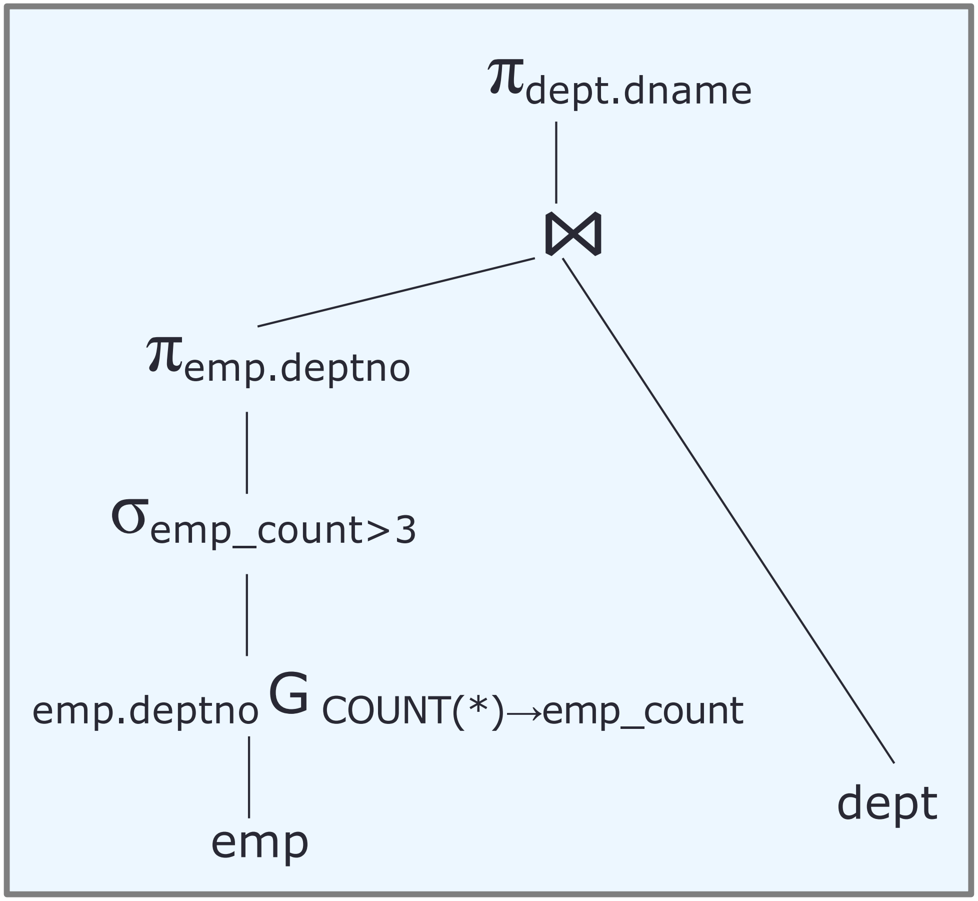
emp(empno, ename, job, mgr, hiredate, salary, comm, deptno)
dept(deptno, dname, loc)

Find the names of departments where more than three employees are working

Another solution:

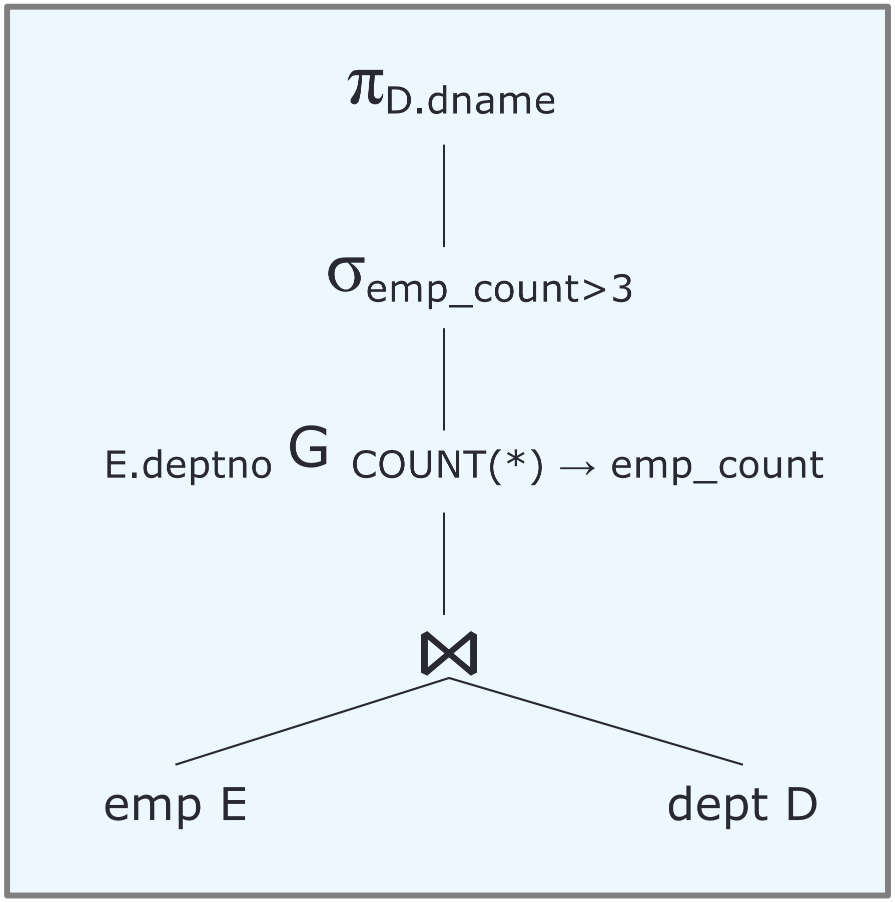
```
SELECT dept.dname
FROM (
   SELECT deptno
   FROM emp
   GROUP BY deptno
   HAVING COUNT(*)>3 )
NATURAL JOIN dept
```

$$\pi_{dept.dname}\ \Big(\ \pi_{emp.deptno}\ \big(\ \sigma_{emp\_count>3}\ \big(\ {}_{emp.deptno}G_{COUNT(*)\rightarrow emp\_count}\ (emp)\ \big)\ \big)\ \bowtie dept\ \Big)$$



Review 2: SQL and RA (solutions 1 vs. 2)

Can you verify equivalence?



$\pi_{D.dname}$

$\sigma_{emp\_count>3}$

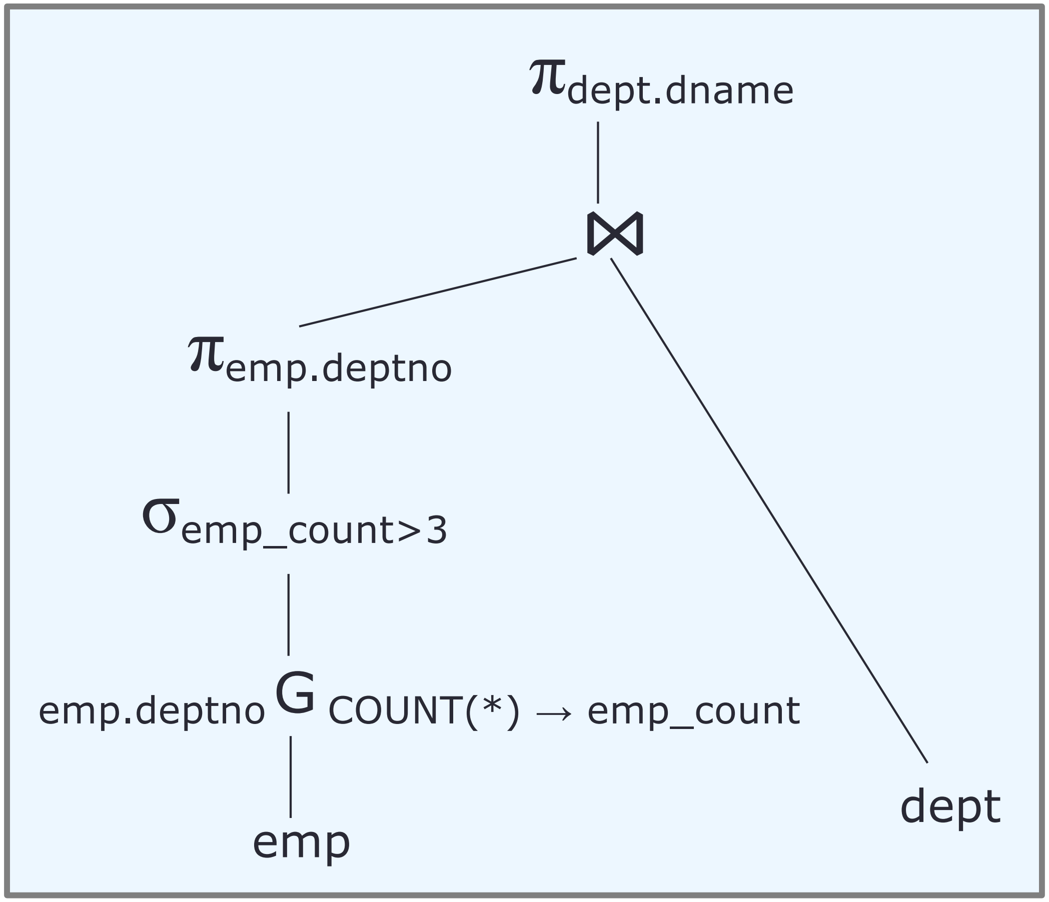
$_{E.deptno}G_{COUNT(*) \rightarrow emp\_count}$

$\bowtie$

emp E

dept D

solution 1



$\pi_{dept.dname}$

$\bowtie$

$\pi_{emp.deptno}$

$\sigma_{emp\_count>3}$

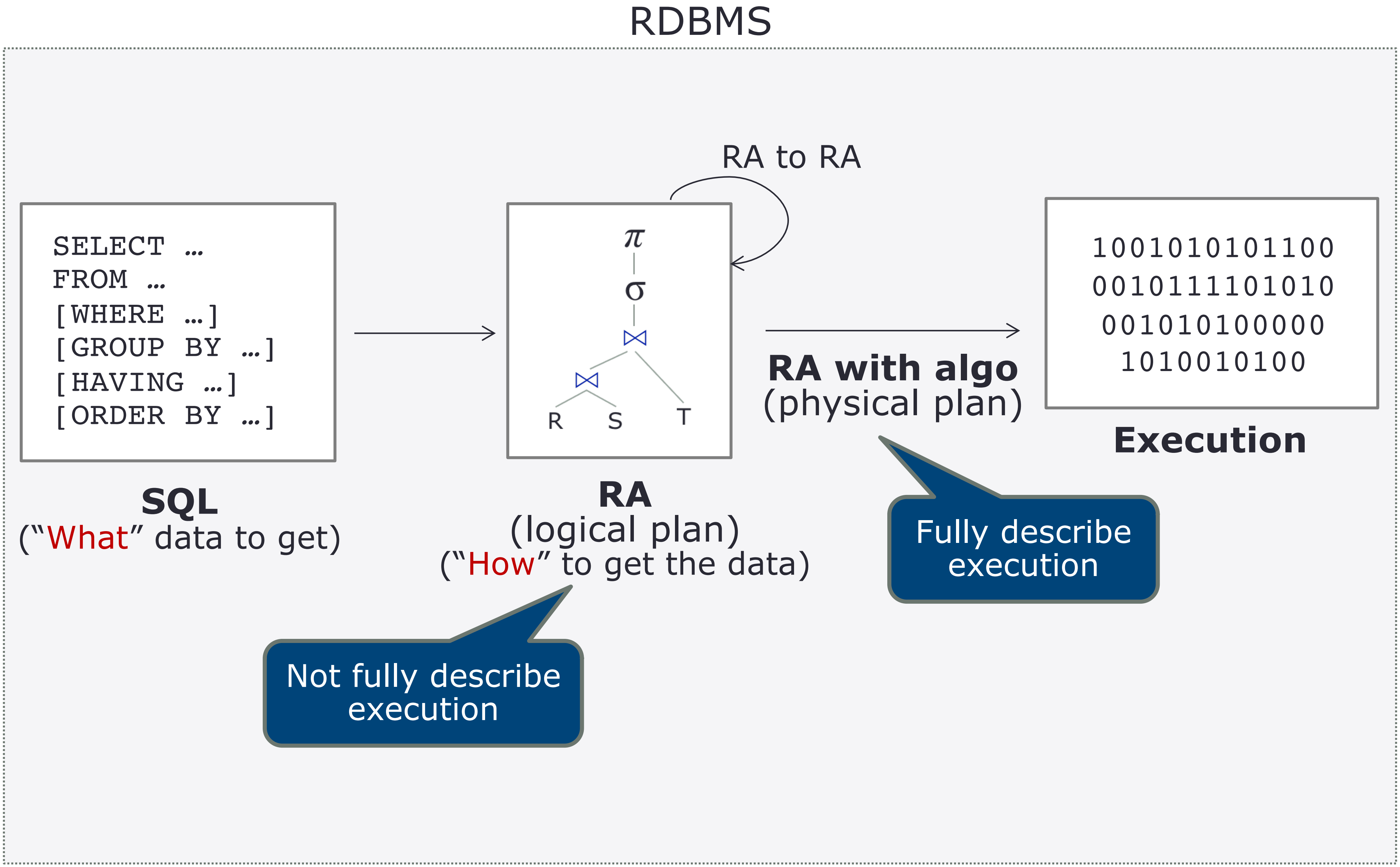
$_{emp.deptno}G_{COUNT(*) \rightarrow emp\_count}$

emp

dept

solution 2

What’s the Point of RA?



RDBMS

```
SELECT …
FROM …
[WHERE …]
[GROUP BY …]
[HAVING …]
[ORDER BY …]
```

SQL
(“What” data to get)

RA
(logical plan)
(“How” to get the data)

Not fully describe execution

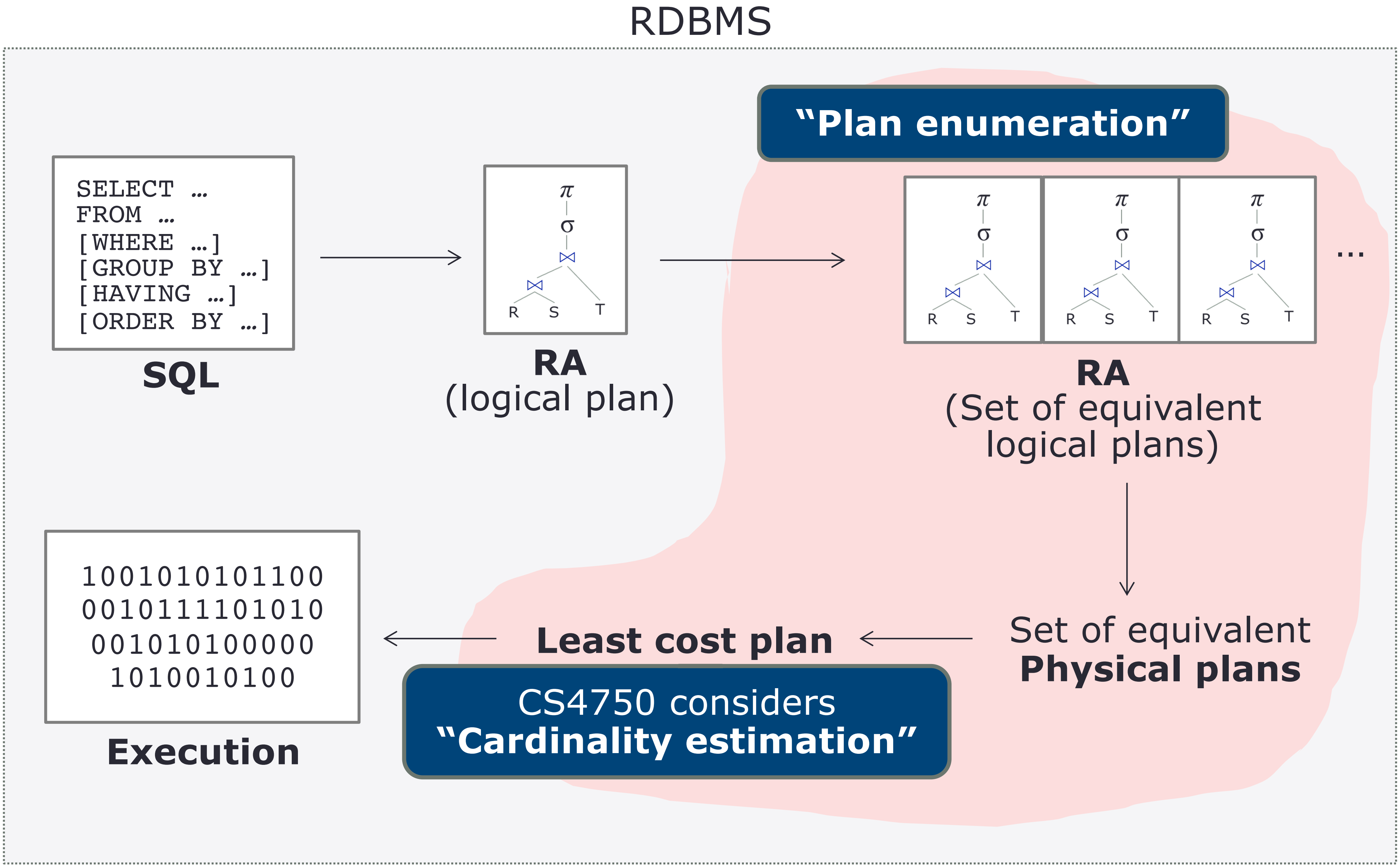
RA with algo
(physical plan)

Fully describe execution

```
1001010101100
0010111101010
001010100000
1010010100
```

Execution

Overview: Query Processing



RDBMS

```
SELECT …
FROM …
[WHERE …]
[GROUP BY …]
[HAVING …]
[ORDER BY …]
```

SQL

RA
(logical plan)

“Plan enumeration”

RA
(Set of equivalent logical plans)

…

Set of equivalent **Physical plans**

Least cost plan

CS4750 considers **“Cardinality estimation”**

```
1001010101100
0010111101010
001010100000
1010010100
```

Execution

Plan Enumeration (RA to RA)

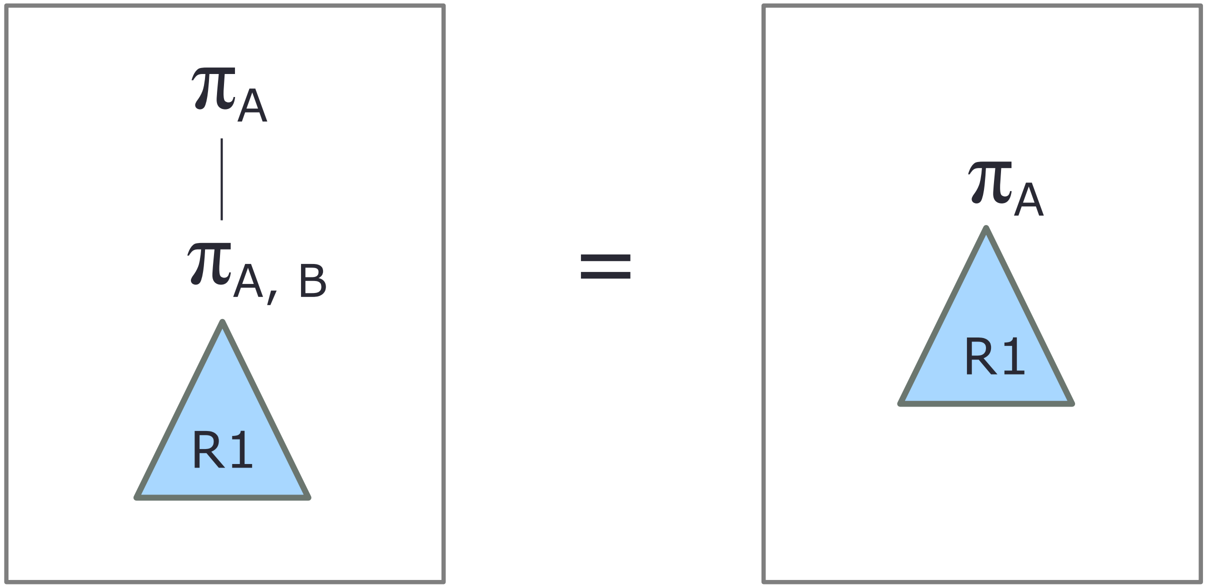
- Some queries can be expressed in different ways

- RA formulation of a query is important in query processing and optimization
  - RA specifies the order of operations
  - The order can largely determine how efficient the query plan will be

- Why RA equivalences?
  - Simplify queries
  - Make queries faster

Explore equivalent RA plans

Simplify "Project"

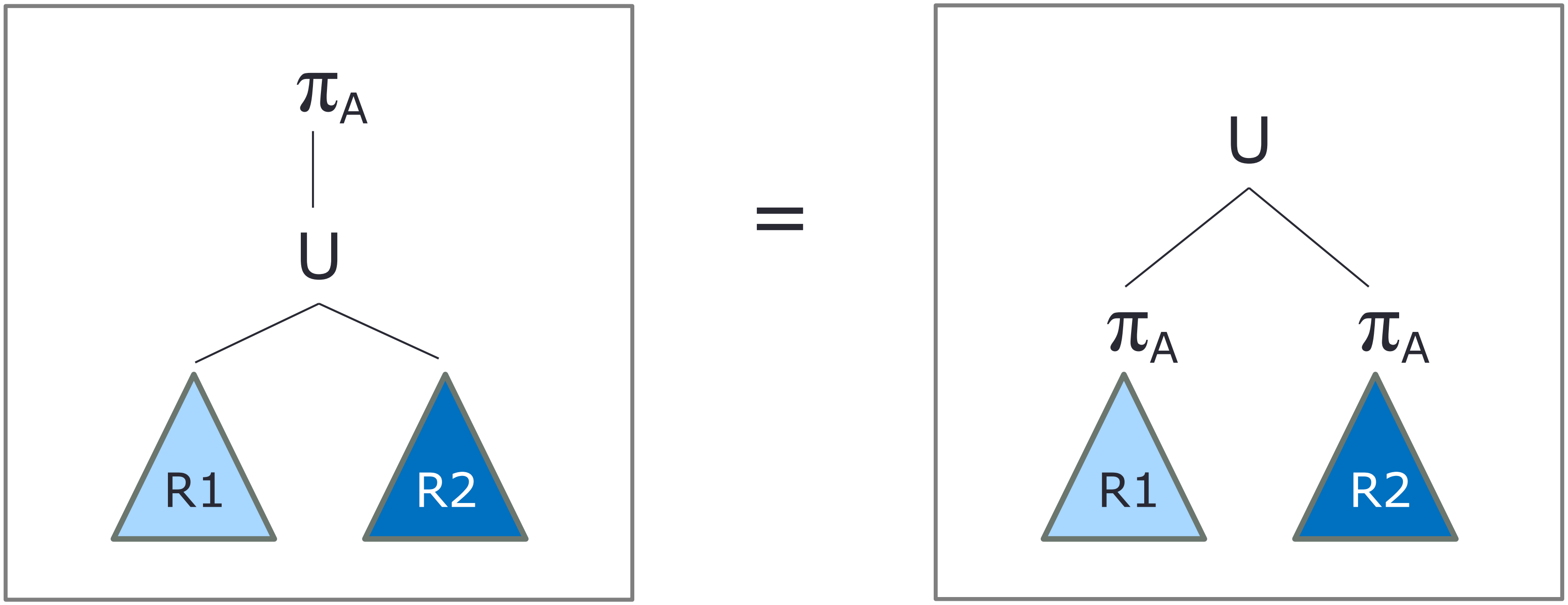


Successive projects can reduced to the final project

only the last project has to be executed

A and *B* are sets of attributes; *R1* is a relation

"Project" and "Union"



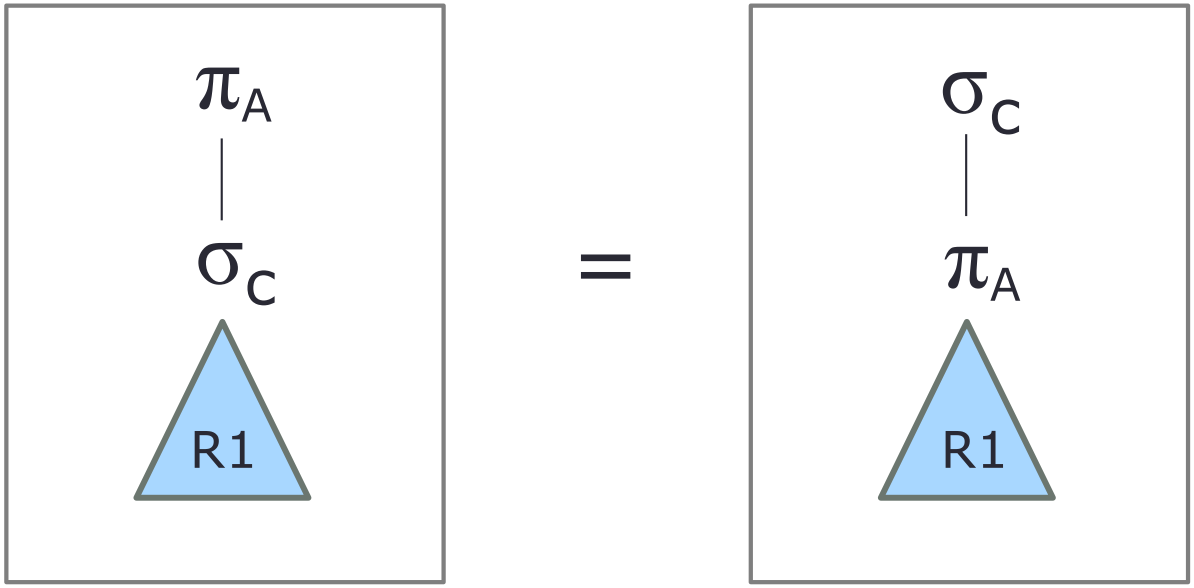
Make sure the schema matches

Note: this may not work with intersection or set difference

Project distributes over union

A is a set of attributes; *R1* and *R2* are relations

“Select” and “Project”

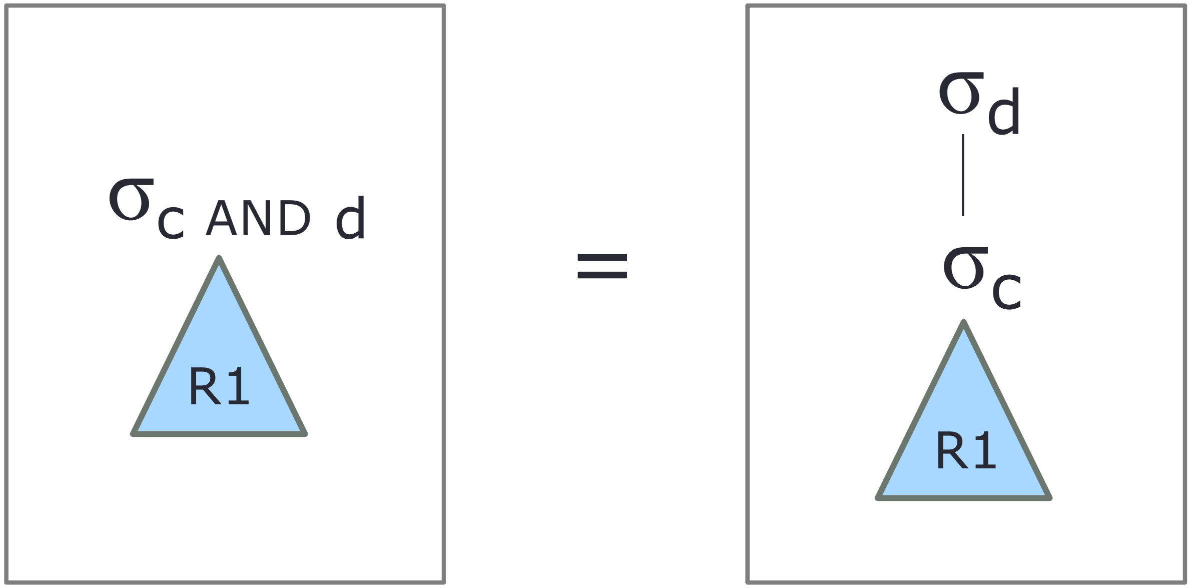


If *C* only references attributes in *A*

Select and project sometimes commute if the condition involves only the attributes in the project list

A is a set of attributes; *C* is a set of Boolean conditions; *R1* is a relation

“Select” and “AND”

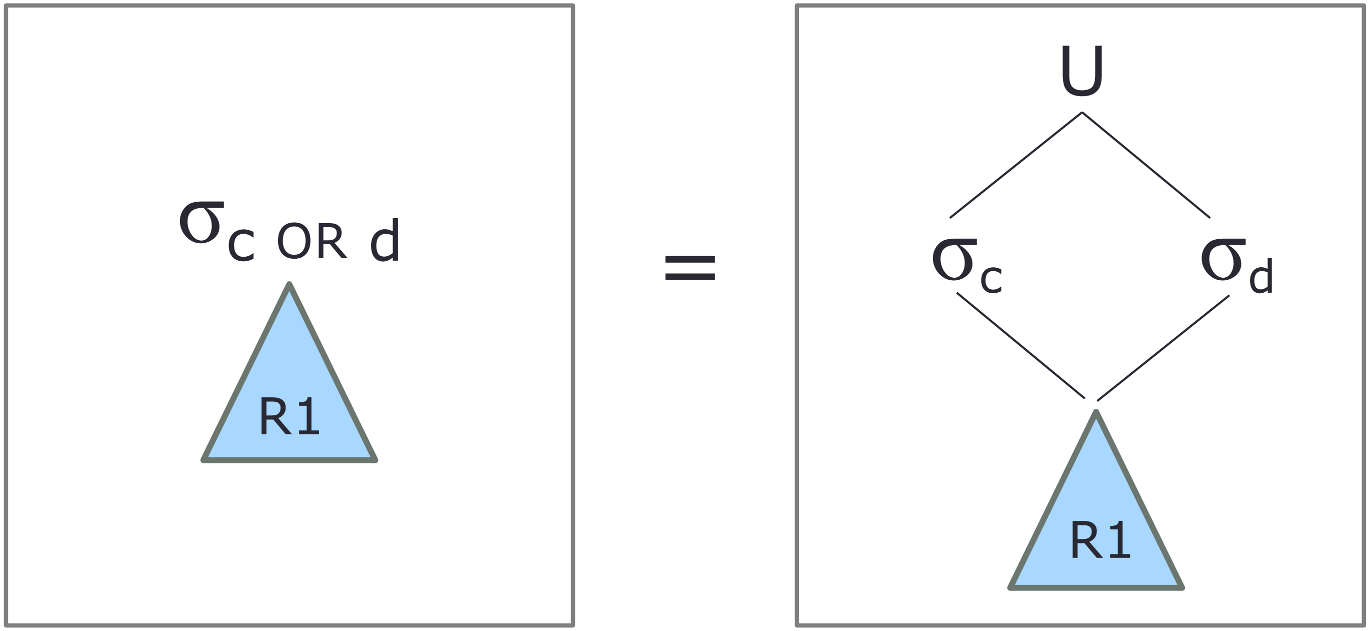


Conjunctive selects can cascade into individual selects

$$\sigma_{c \text{ AND } d} (R1)) \quad = \quad \sigma_d(\sigma_c(R1))$$

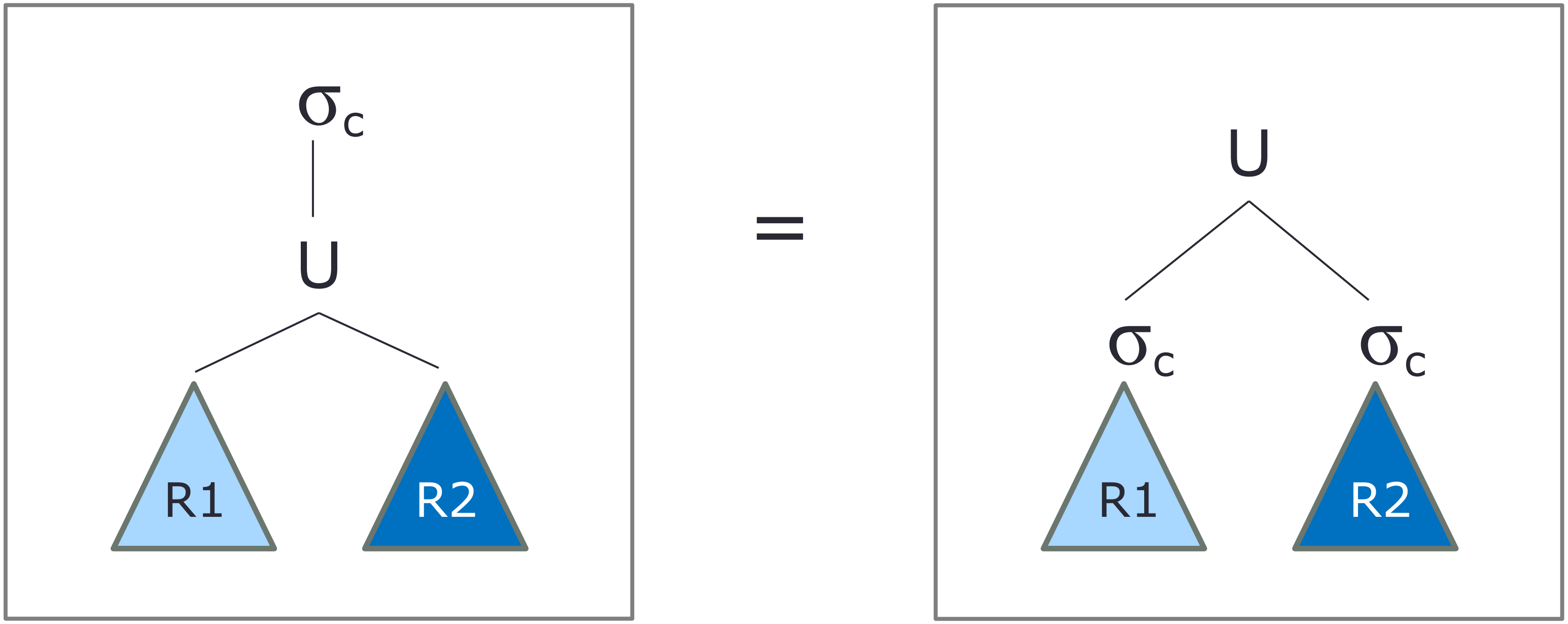
c and *d* are Boolean conditions; *R1* is a relation

“Select” and “OR”



c and *d* are Boolean conditions; *R1* is a relation

"Select" and "Union"

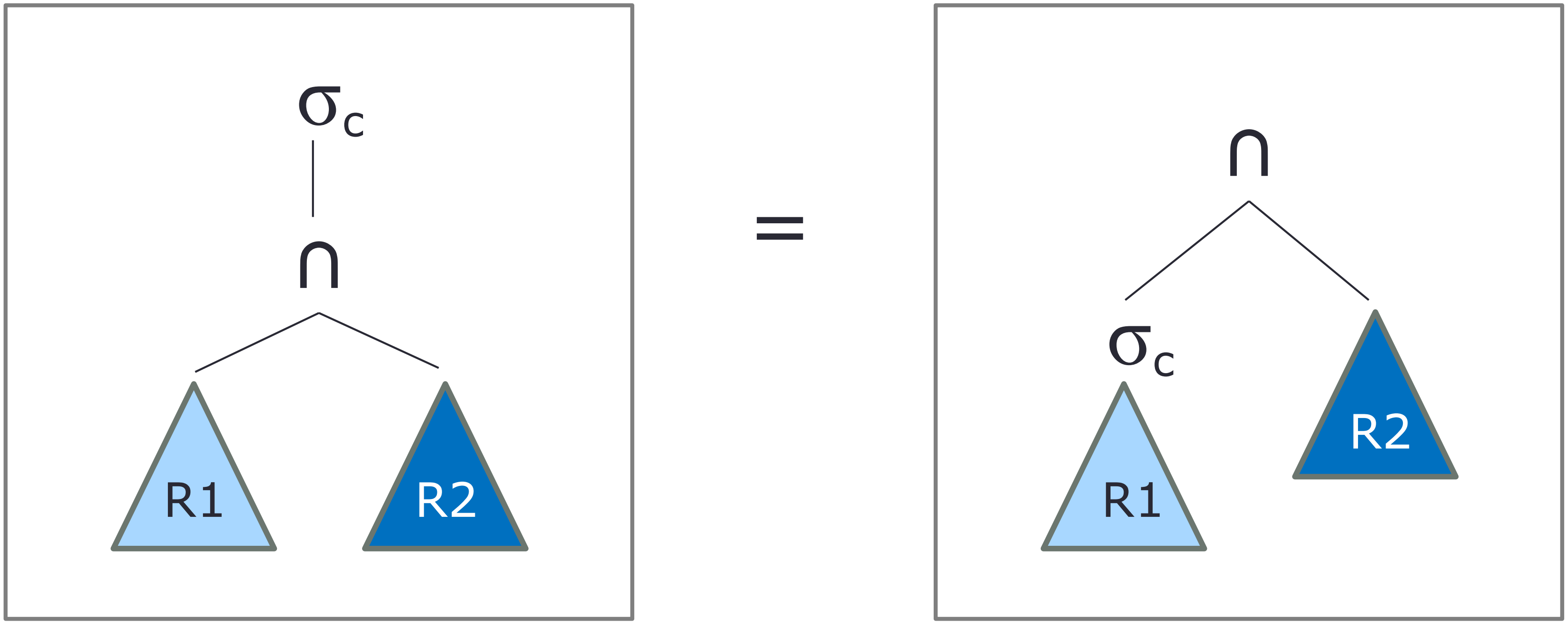


If *C* references attributes in *R1* and *R2*

Perform "Select" operation as early as possible

C is a set of Boolean conditions; *R1* and *R2* are relations

“Select” and “Intersect”

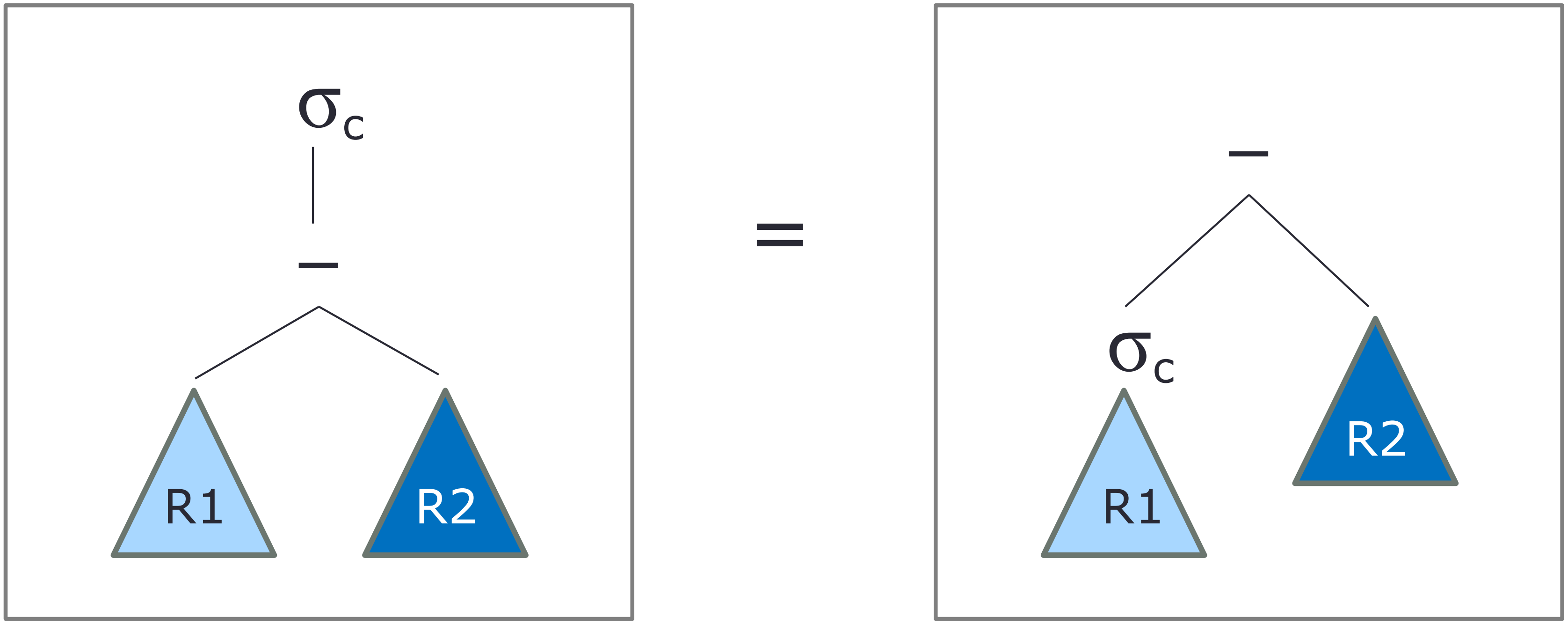


If *C* only references attributes in *R1*

Perform “Select” operation as early as possible

C is a set of Boolean conditions; *R1* and *R2* are relations

“Select” and “Difference”



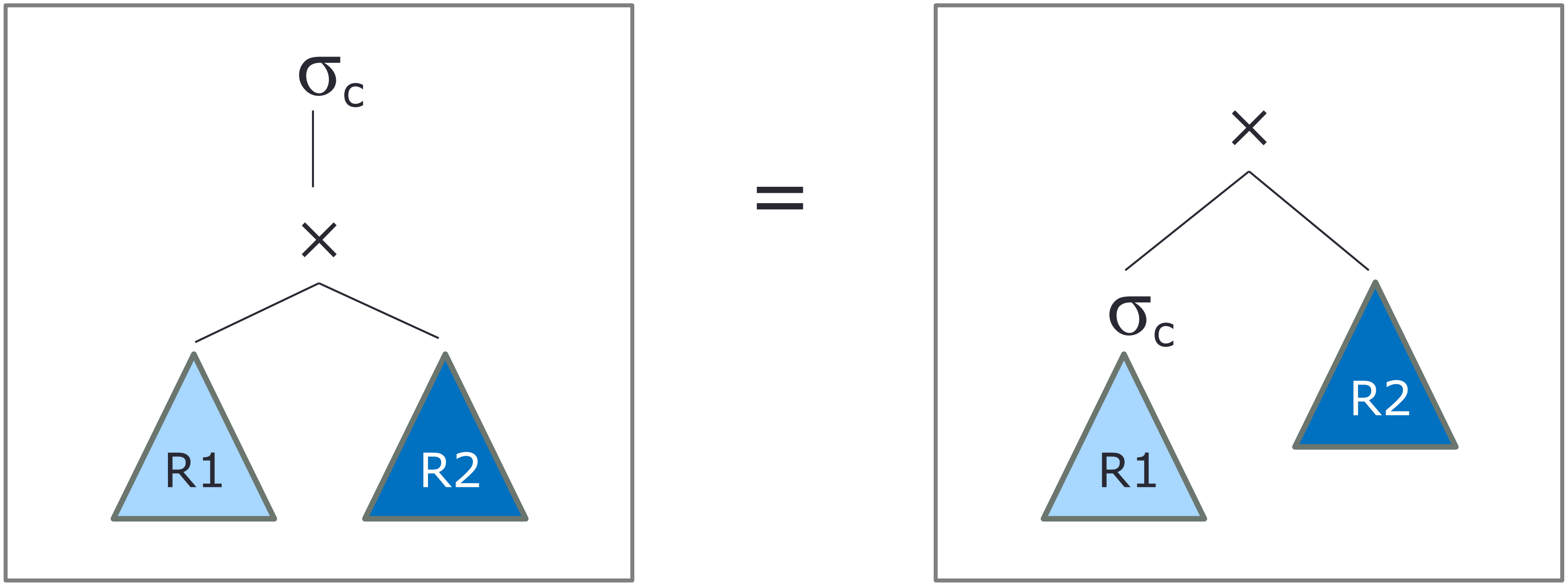
If *C* only references attributes in *R1*

Perform “Select” operation as early as possible

“EXCEPT” in SQL is equivalent to set “difference” in RA

C is a set of Boolean conditions; *R1* and *R2* are relations

“Select” and “Cartesian Product”



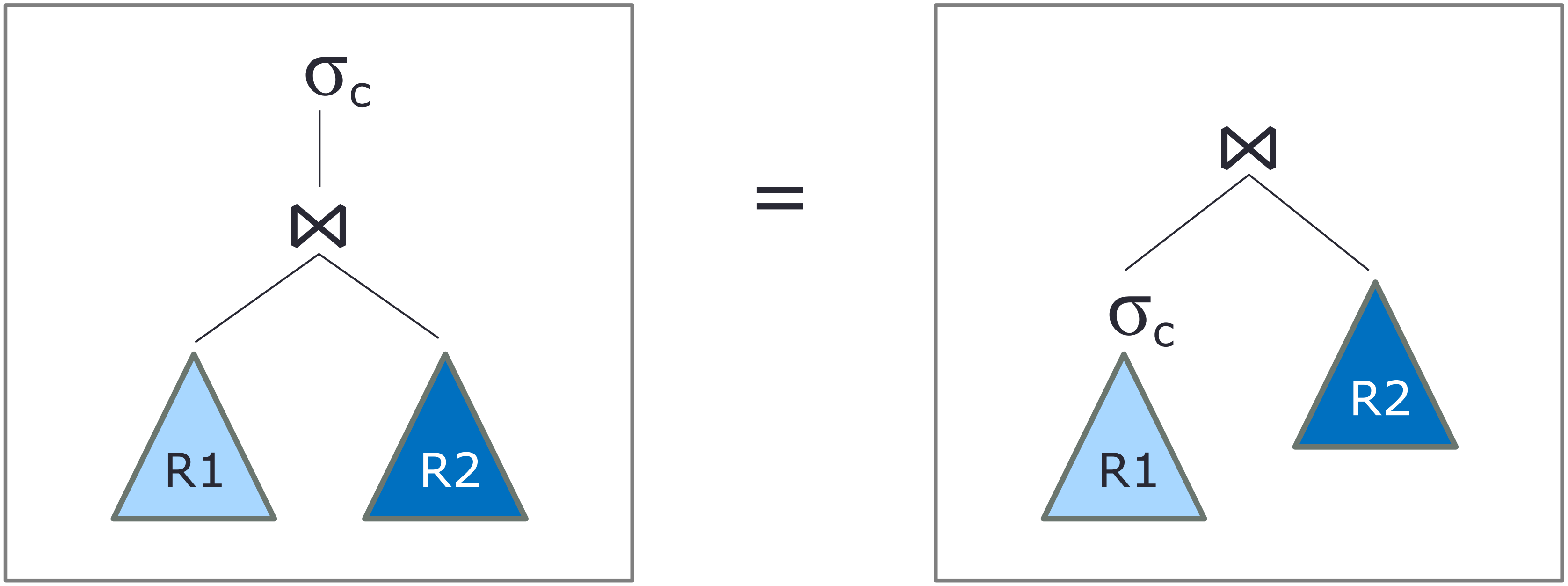
If *C* only references attributes in *R1*

Select and product sometimes commute

Perform “Select” operation as early as possible

C is a set of Boolean conditions; *R1* and *R2* are relations

"Select" and "Join"



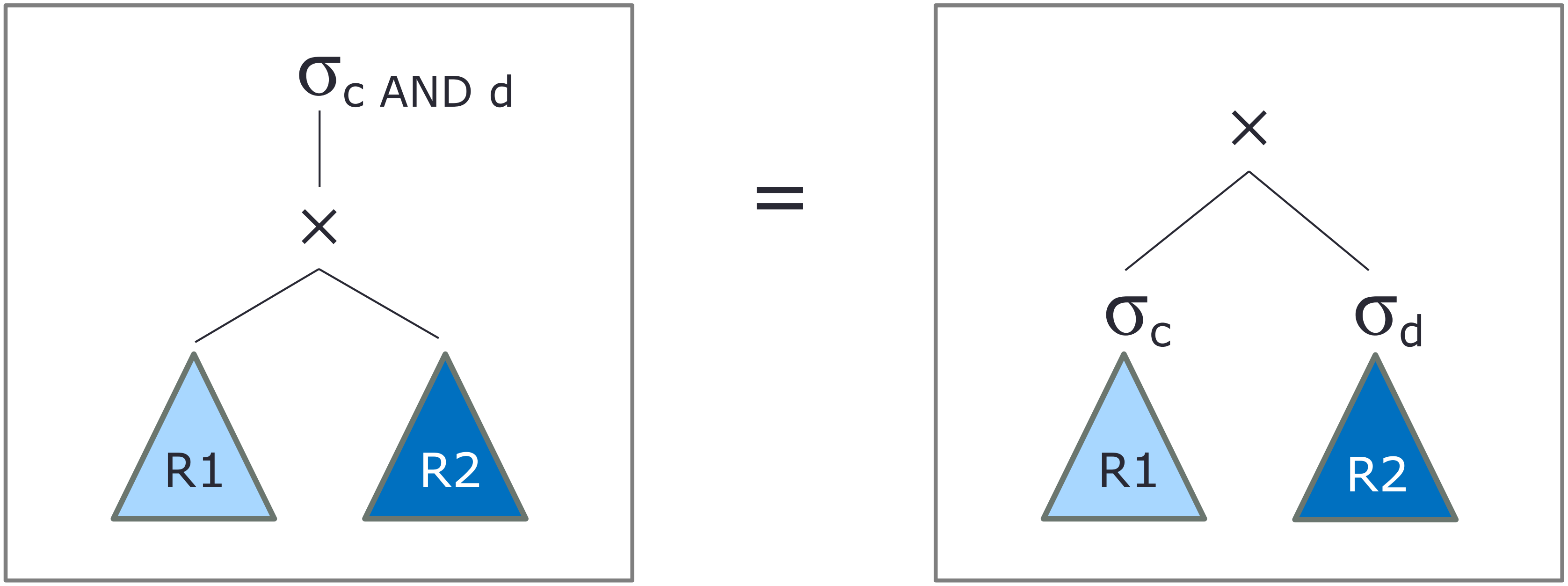
If *C* only references attributes in *R1*

Select and join sometimes commute

Perform "Select" operation as early as possible

C is a set of Boolean conditions; *R1* and *R2* are relations

“Select” and “Cartesian Product”



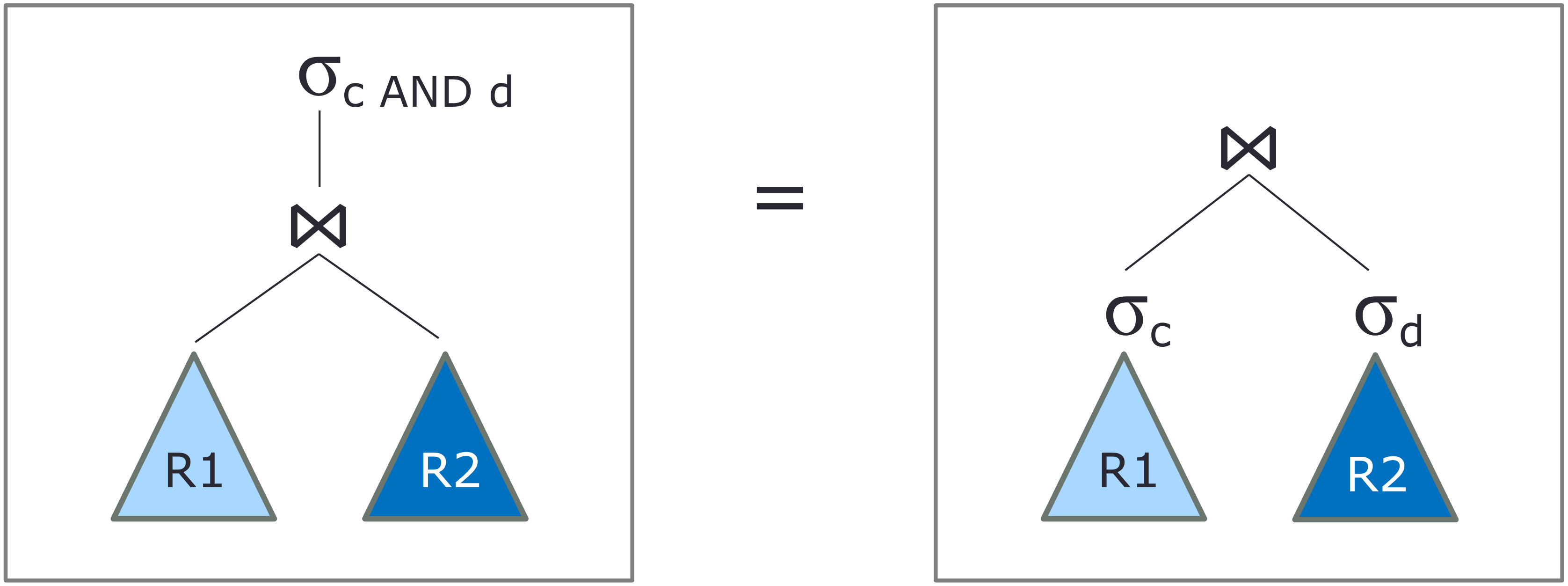
If *c* references attributes in *R1* and *d* references attributes in *R2*

Select sometimes distribute over product

Perform “Select” operation as early as possible

C is a set of Boolean conditions; *R1* and *R2* are relations

“Select” and “Join”



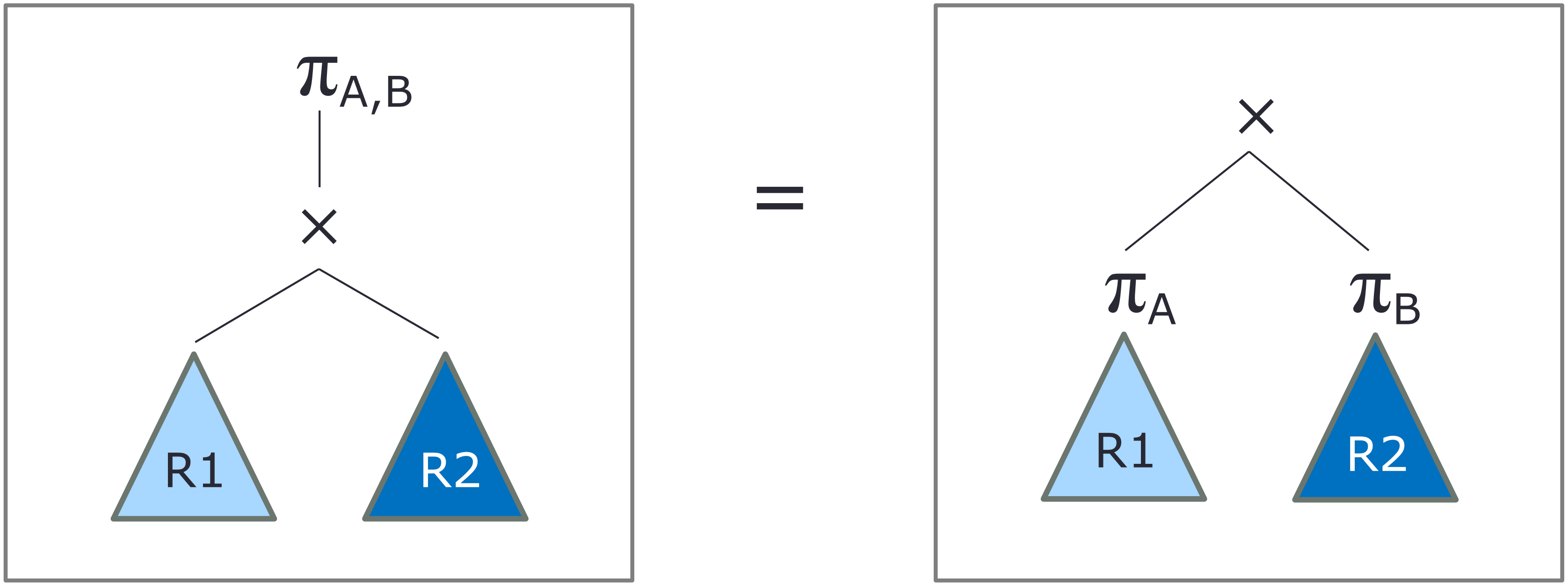
If *c* references attributes in *R1* and *d* references attributes in *R2*

Select sometimes distribute over join

Perform “Select” operation as early as possible

C is a set of Boolean conditions; *R1* and *R2* are relations

“Project” and “Cartesian Product”

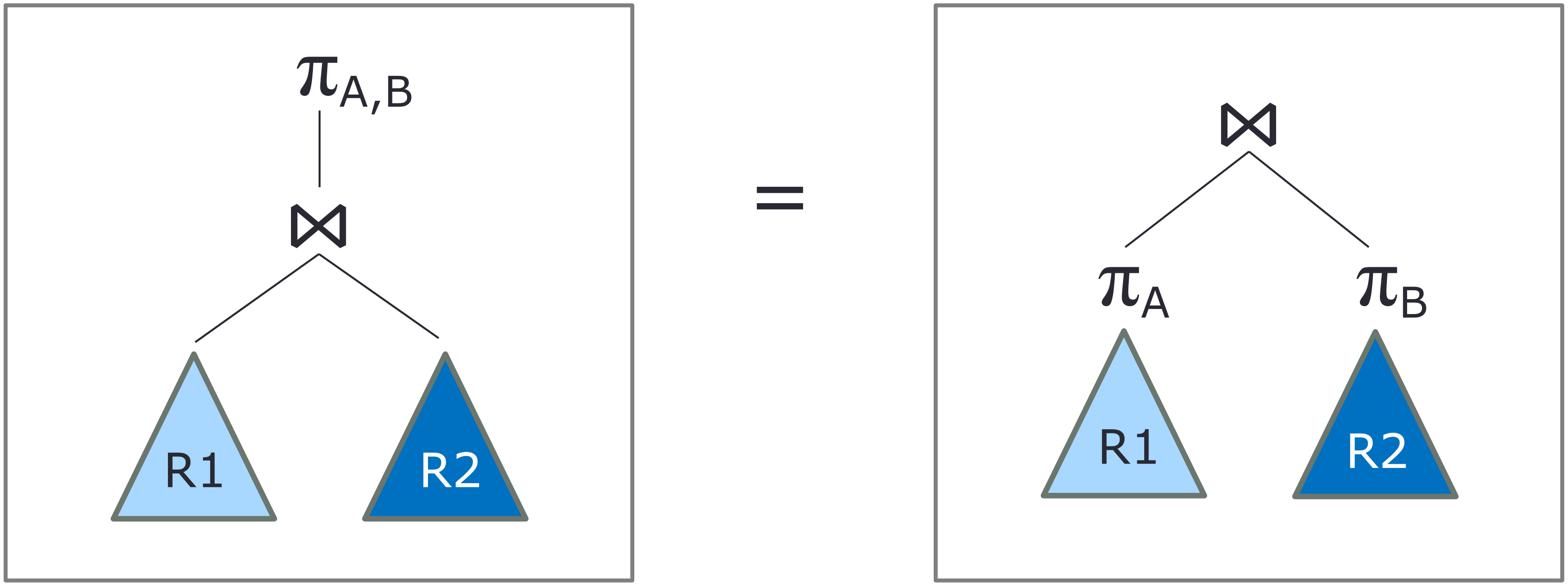


If *A* contains attributes in *R1* and *B* contains attributes in *R2*

Project sometimes distribute
over product

A and *B* are sets of attributes; *R1* and *R2* are relations

“Project” and “Join”

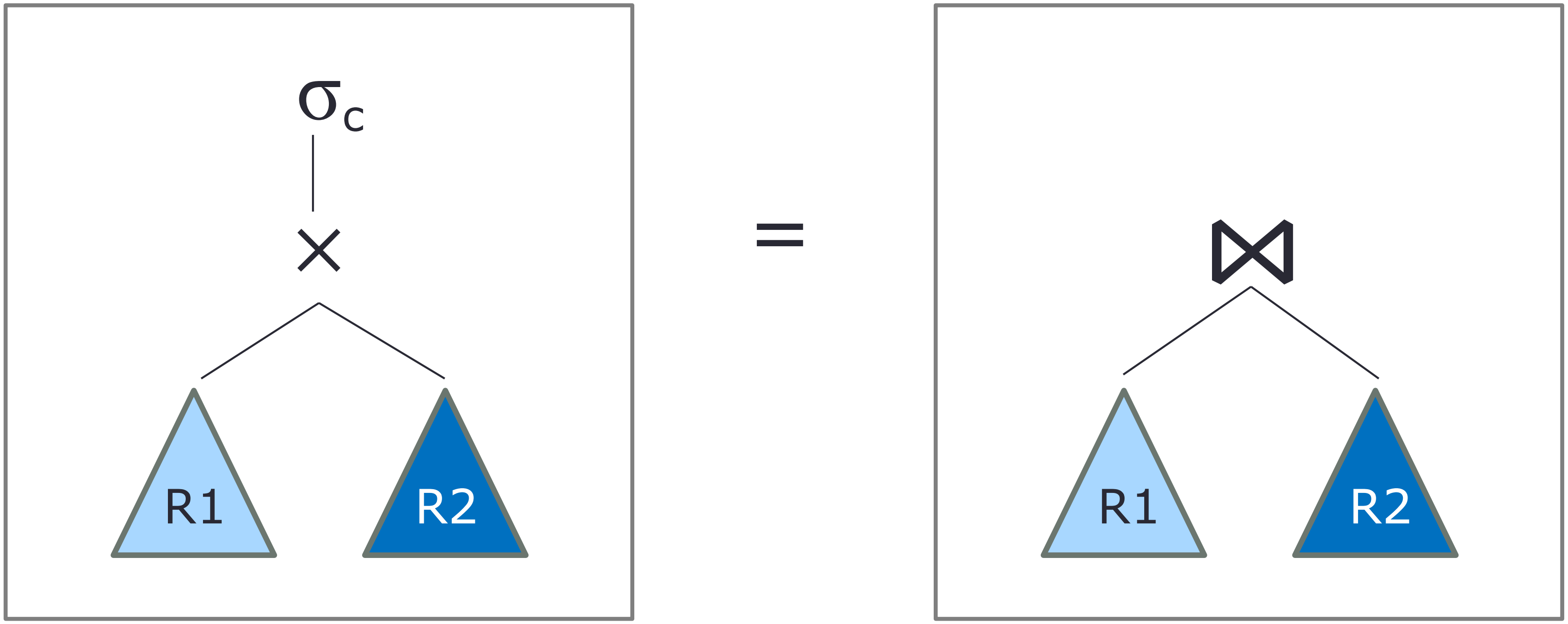


If *A* contains attributes in *R1* and *B* contains attributes in *R2*

Project sometimes distribute
over join

A and *B* are sets of attributes; *R1* and *R2* are relations

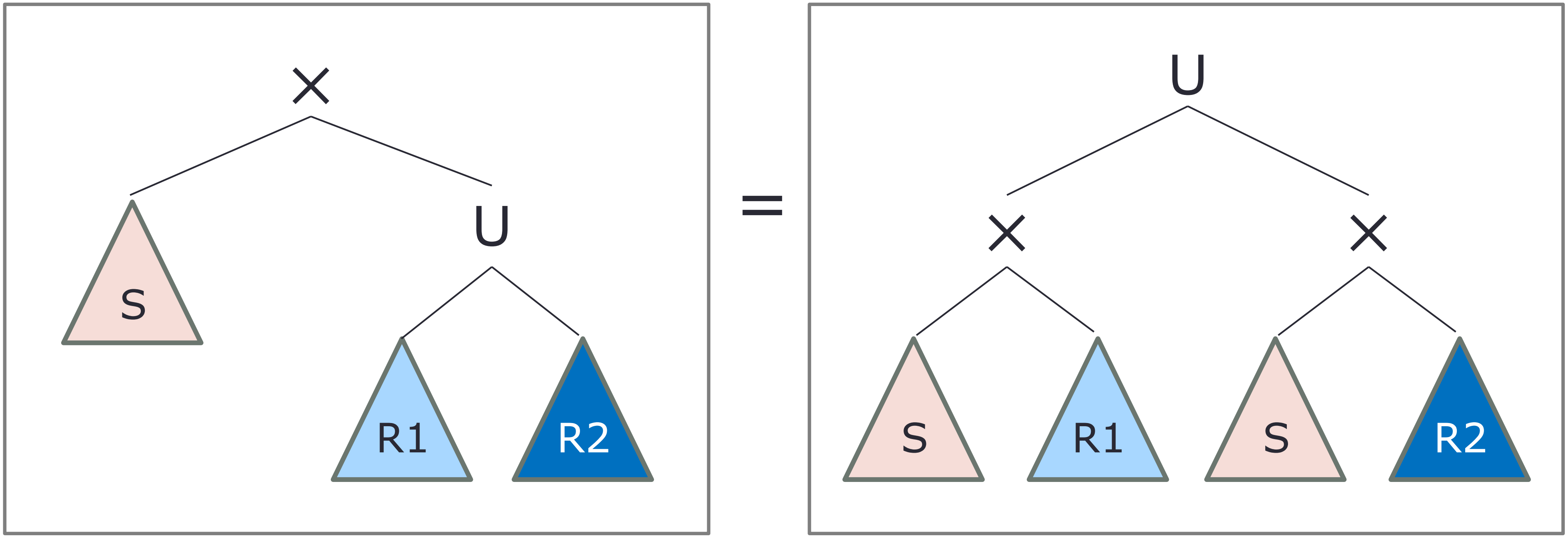
“Cartesian Product” and “Join”



Assume *C* contains attributes that are used to combine relations

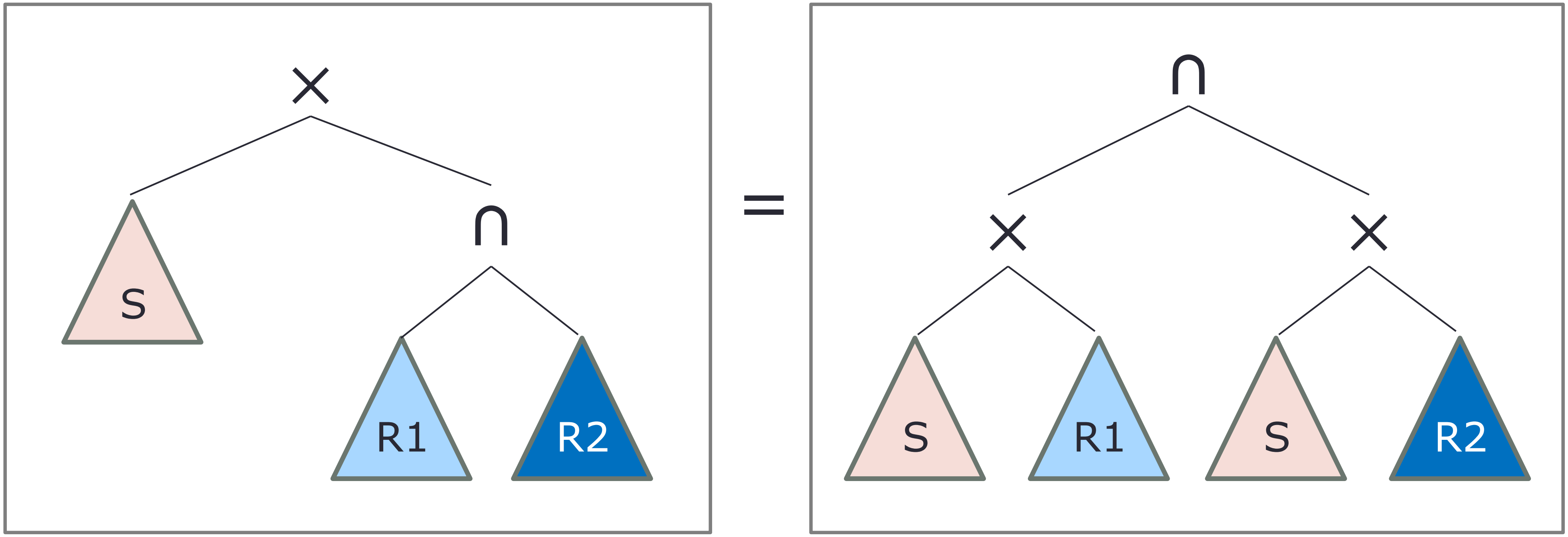
C is a set of Boolean conditions; *R1* and *R2* are relations

“Cartesian Product” and “Union”



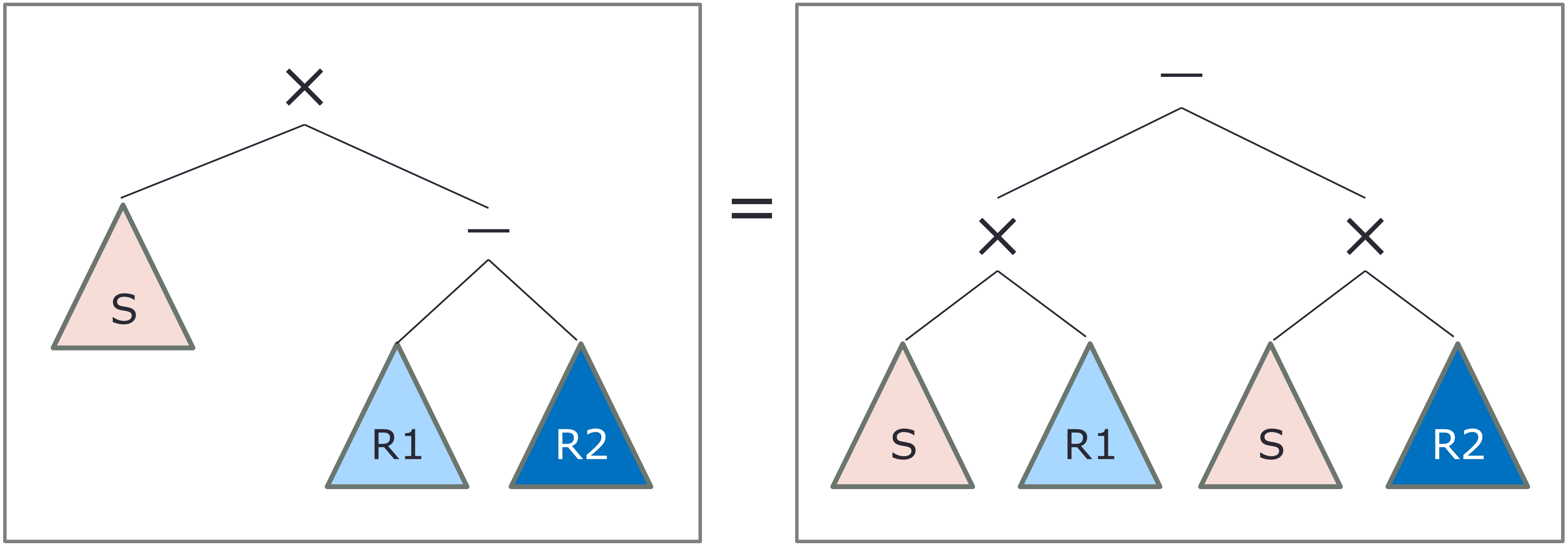
R1, *R2*, and *S* are relations

“Cartesian Product” and “ Intersect”



R1, *R2*, and *S* are relations

“Cartesian Product” and “Difference”



“EXCEPT” in SQL is equivalent to set difference in RA

R1, *R2*, and *S* are relations

More RA Equivalences

- All joins and Cartesian products are commutative

  $R \times S = S \times R$ (mostly)

  $R \bowtie S = S \bowtie R$

- Joins and Cartesian products are associative

  $(R \times S) \times T = R \times (S \times T)$

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

- Select is commutative

  $\sigma_c(\sigma_d(R)) = \sigma_d(\sigma_c(R))$

- Union and intersection are commutative

  $R \cup S = S \cup R$

  $R \cap S = S \cap R$

- Union and intersection are associative

  $(R \cup S) \cup T = R \cup (S \cup T)$

  $(R \cap S) \cap T = R \cap (S \cap T)$

Plenty more equivalences

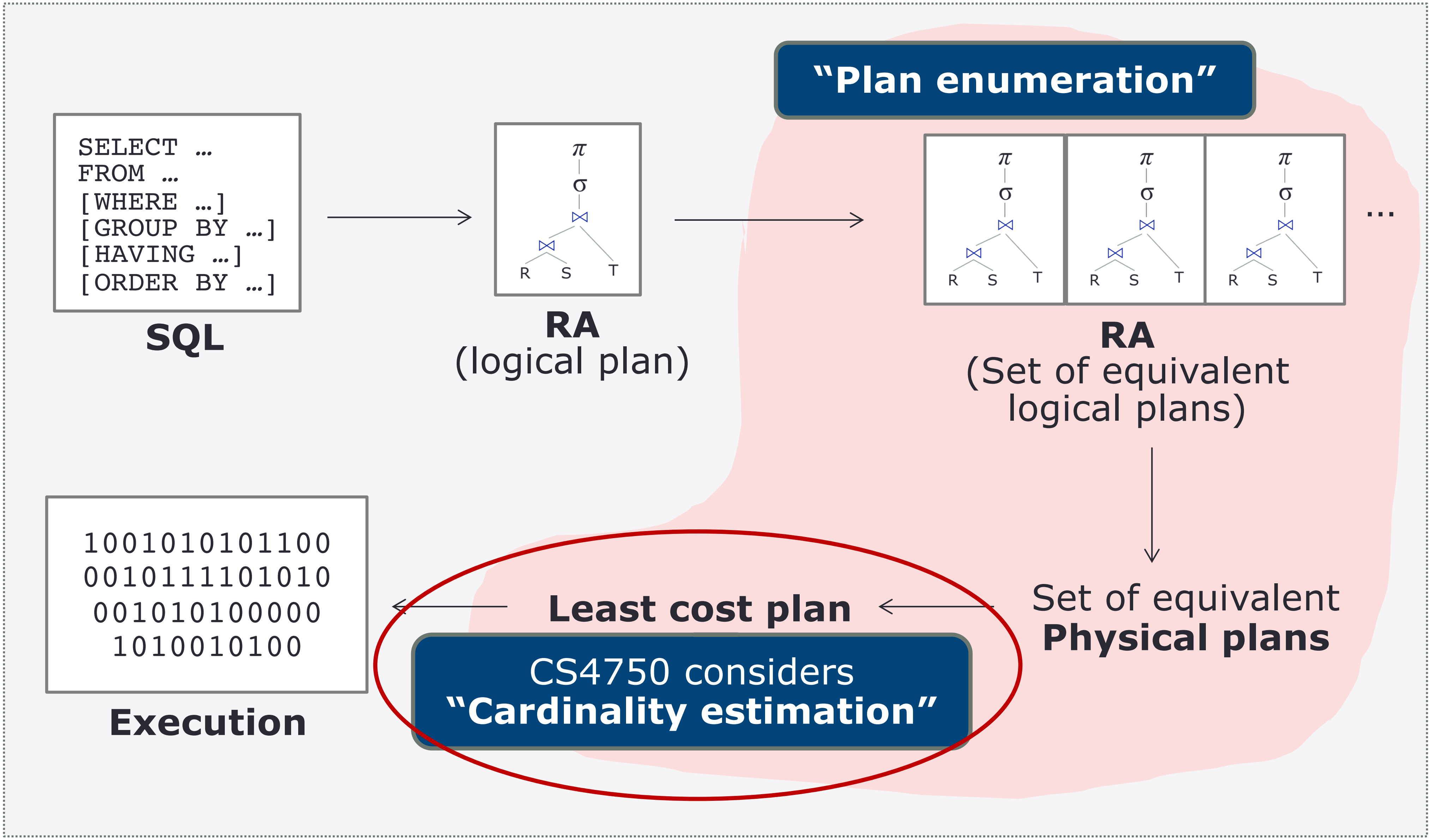
How to remember?
Use the definitions

R, *S*, and *T* are relations

Overview: Query Optimization

revisit

RDBMS



```
SELECT …
FROM …
[WHERE …]
[GROUP BY …]
[HAVING …]
[ORDER BY …]
```

SQL

RA
(logical plan)

"Plan enumeration"

RA
(Set of equivalent logical plans)

Set of equivalent
Physical plans

Least cost plan

CS4750 considers
"Cardinality estimation"

```
1001010101100
0010111101010
001010100000
1010010100
```

Execution

Disclaimer

Cost estimation is an active research topic.

Equations and methods discussed here form a foundation of concepts, but usually cannot compare to a commercialized solution.

General Idea on Plan Selection

- Which equivalent RA leads to the most efficient algorithm?
- What algorithm should we use to implement each operation?
- How should the operations pass data from one to the other?

Depends on info available to the query optimizer

- Size of each relation
- Statistics (#blocks, #tuples, #distinct values for an attribute)
- Indexes
- Layout of data on disk

For this class, we assume

- Disk-based storage – HDD
- Row-based storage – tuples are stored contiguously
- HDD I/O cost (reading from disk) only considered
- Sequential disk reads
- A block can be read at once
- No data preloaded

Cardinality Estimation

Estimate the **number of tuples** in the output of each RA operator

Let’s use the University database schema as a running example

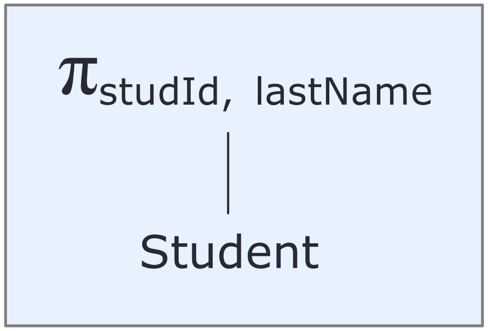
```
Student (studId, lastName, fistName, major, credits)
Class (classNumber, facId, schedule, room)
Faculty (facId, name, department, rank)
Enroll (studId, classNumber, grade)
```

Estimation: SELECT

Assume we know the following information:

```
Student (studId, lastName, fistName, major, credits)
  T(Student) = 10,000                 #of tuples
  V(lastName) = 9,500                 #of distinct values
  V(major) = 10                       #of distinct values
  Range(credits) = [1, 126)           range of values
```

```
SELECT studId, lastName
FROM Student
```



$\pi_{\text{studId, lastName}}$

Student

How many tuples do we expect this query to output?

10,000

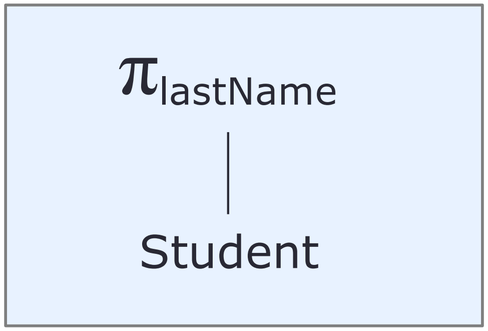
Estimation: DISTINCT

Assume we know the following information:

`Student (studId, lastName, fistName, major, credits)`

`T(Student) = 10,000`	#of tuples
`V(lastName) = 9,500`	#of distinct values
`V(major) = 10`	#of distinct values
`Range(credits) = [1, 126)`	range of values

```
SELECT DISTINCT lastName
FROM Student
```



$\pi_{lastName}$

Student

How many tuples do we expect this query to output?

9,500

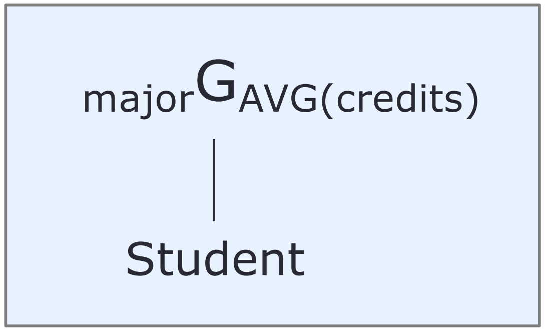
Estimation: AGGREGATE

Assume we know the following information:

Student (<u>studId</u>, lastName, fistName, major, credits)

`T(Student) = 10,000`	#of tuples
`V(lastName) = 9,500`	#of distinct values
`V(major) = 10`	#of distinct values
`Range(credits) = [1, 126)`	range of values

```
SELECT major, AVG(credits)
FROM Student
GROUP BY major
```



$_{major}G_{AVG(credits)}$

Student

How many tuples do we expect this query to output?

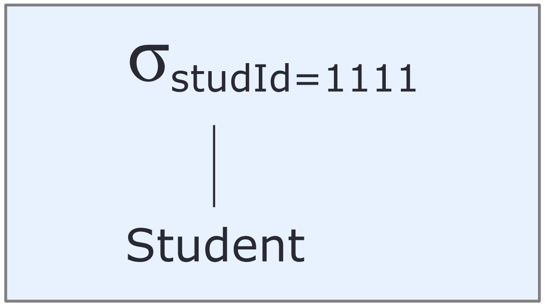
10

Estimation: WHERE Value

Assume we know the following information:

```
Student (studId, lastName, fistName, major, credits)
  T(Student) = 10,000                 #of tuples
  V(lastName) = 9,500                 #of distinct values
  V(major) = 10                       #of distinct values
  Range(credits) = [1, 126)           range of values
```

```
SELECT *
FROM Student
WHERE studId = 1111
```



How many tuples do we expect this query to output?
(assume that 1111 exists)

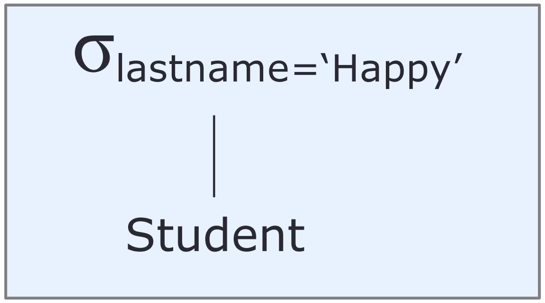
1

Estimation: WHERE Value

Assume we know the following information:

```
Student (studId, lastName, fistName, major, credits)
  T(Student) = 10,000                    #of tuples
  V(lastName) = 9,500                    #of distinct values
  V(major) = 10                          #of distinct values
  Range(credits) = [1, 126)              range of values
```

```
SELECT *
FROM Student
WHERE lastname = ’Happy’
```



$\sigma_{lastname=‘Happy’}$

Student

How many tuples do we expect this query to output?
(assume distinct values uniformly distribute; constant ‘Happy’ exists; without assumption, estimate is impossible)

$$T(R) * \underbrace{\boxed{\frac{1}{V(attr)}}}_{\text{Selectivity factor}} = 10000 * \frac{1}{9500} \approx 1.05 \text{ tuples}$$

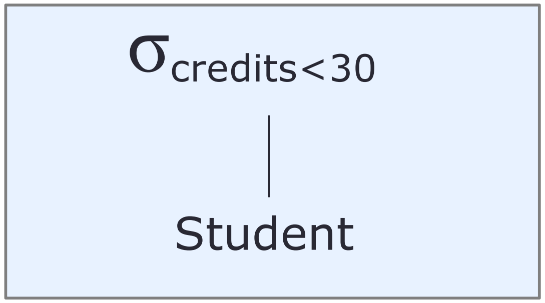
Estimation: WHERE Range

Assume we know the following information:

Student (studId, lastName, fistName, major, credits)

T(Student) = 10,000	#of tuples
V(lastName) = 9,500	#of distinct values
V(major) = 10	#of distinct values
Range(credits) = [1, 126)	range of values

```
SELECT *
FROM Student
WHERE credits < 30
```



$\sigma_{credits<30}$

Student

How many tuples do we expect this query to output?
(assume uniformly distributed and continuous; without assumption, estimate is impossible)

$$T(R) * \boxed{\frac{(\text{val} - \min)}{(\max - \min)}} = 10000 * \frac{(30-1)}{(126-1)} \approx 2320 \text{ tuples}$$

Selectivity factor = (selection range) / (total range)

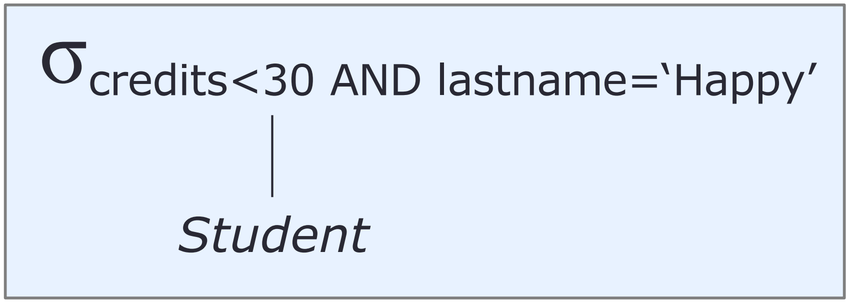
Estimation: AND

Assume we know the following information:

```
Student (studId, lastName, fistName, major, credits)
```

`T(Student) = 10,000`	#of tuples
`V(lastName) = 9,500`	#of distinct values
`V(major) = 10`	#of distinct values
`Range(credits) = [1, 126)`	range of values

```
SELECT *
FROM Student
WHERE credits < 30
AND lastname = ‘Happy’
```



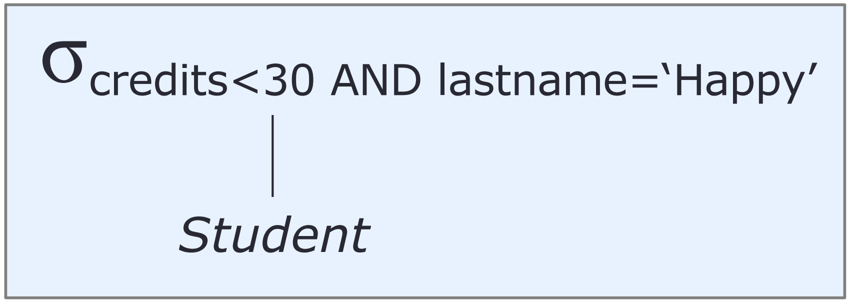
$\sigma_{\text{credits<30 AND lastname=`Happy'}}$

Student

How many tuples do we expect this query to output?
(assume constants exist, distinct values uniformly distributed and continuous, ‘Happy’ exists; without assumption, estimate is impossible)

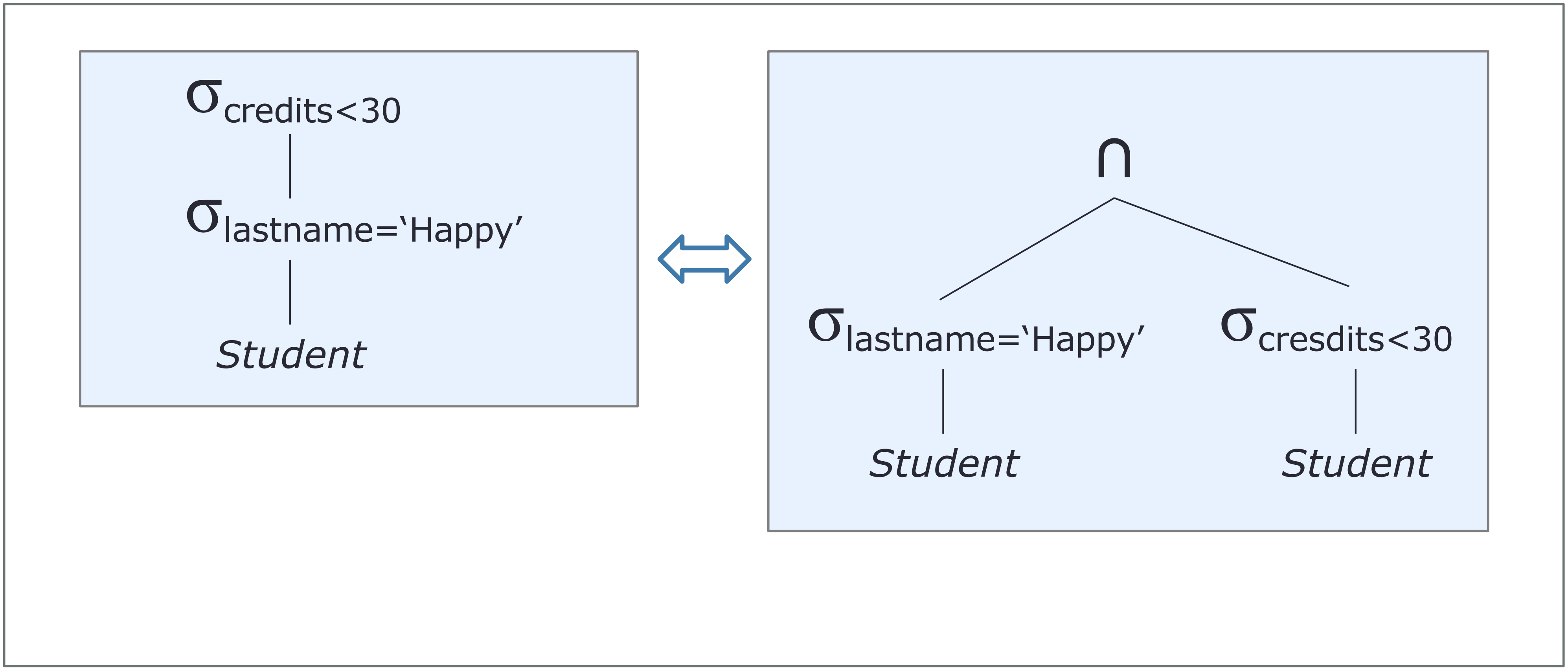
Estimation: AND (2)

```
SELECT *
FROM Student
WHERE credits < 30
AND lastname = 'Happy'
```



$\sigma_{\text{credits<30 AND lastname='Happy'}}$

Student



$\sigma_{\text{credits<30}}$

$\sigma_{\text{lastname='Happy'}}$

Student

⇔

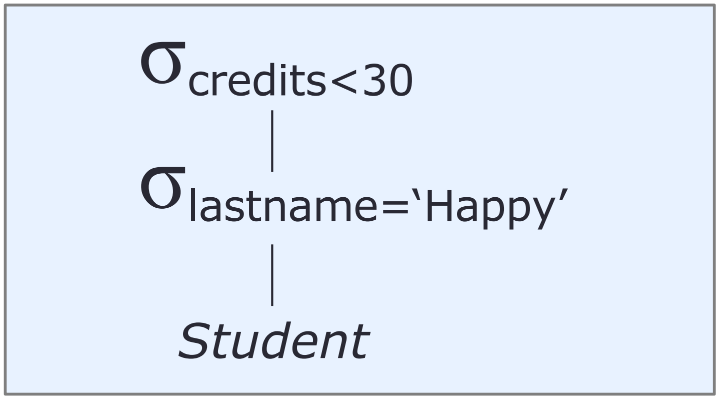
∩

$\sigma_{\text{lastname='Happy'}}$ $\sigma_{\text{cresdits<30}}$

Student *Student*

Estimation: AND (3)

```
SELECT *
FROM Student
WHERE credits < 30
AND lastname = ‘Happy’
```



$\sigma_{\text{credits<30}}$

$\sigma_{\text{lastname=‘Happy’}}$

Student

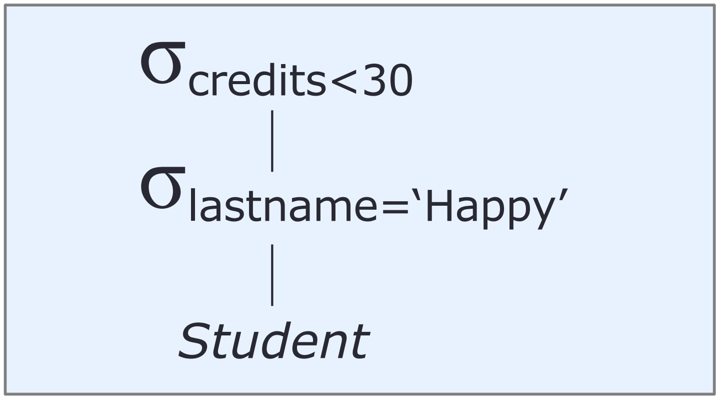
How many tuples do we expect this query to output? – hard to say

- If conditions disjoint, 0 tuple result
  - E.g. no student with lastname ‘Happy’ has credits <30
- If conditions independent, there will be multiple estimates
  - E.g. lastname and credits are independent
- If conditions fully overlap, take minimum of estimates
  - E.g. all students with lastname ‘Happy’ have credits <30

Assume independent unless you know for sure full overlap

Estimation: AND (4)

```
SELECT *
FROM Student
WHERE credits < 30
AND lastname = 'Happy'
```



$\sigma_{\text{credits<30}}$

$\sigma_{\text{lastname='Happy'}}$

Student

How many tuples do we expect this query to output? – hard to say

- If conditions disjoint, 0 tuple result

= 0

Selectivity factor

- If conditions independent, there will be multiple estimates

≈ 10000 * ((30-1) / (126-1)) * (1/9500) ≈ 0.244 tuples

- If conditions fully overlap, take minimum of estimates

≤ 10000 * min{ ((30-1) / (126-1)), (1/9500) } ≈ 1.053 tuples

(assume independent unless otherwise specified --- answer: 0.244 tuples)

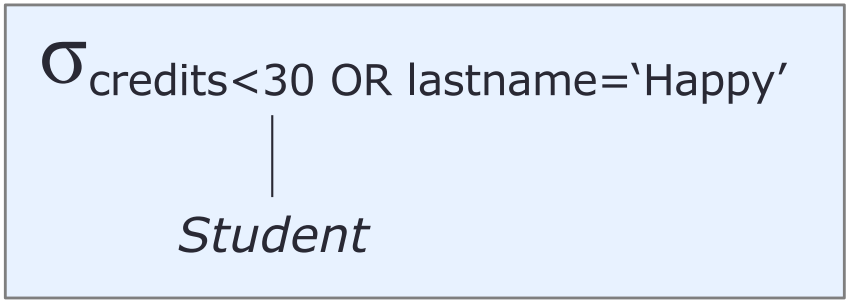
Estimation: OR

Assume we know the following information:

```
Student (studId, lastName, fistName, major, credits)
```

`T(Student) = 10,000`	#of tuples
`V(lastName) = 9,500`	#of distinct values
`V(major) = 10`	#of distinct values
`Range(credits) = [1, 126)`	range of values

```
SELECT *
FROM Student
WHERE credits < 30
OR lastname = ‘Happy’
```



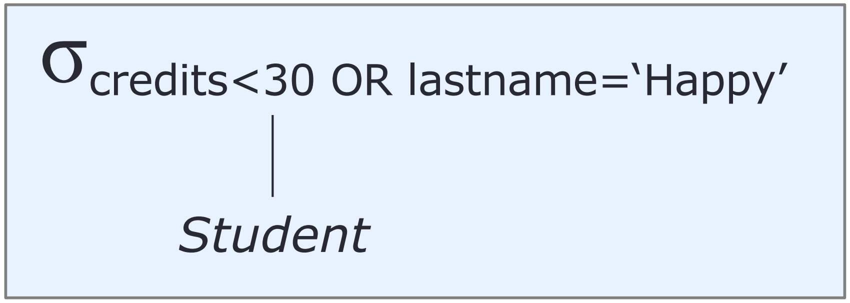
$\sigma_{\text{credits<30 OR lastname=‘Happy’}}$

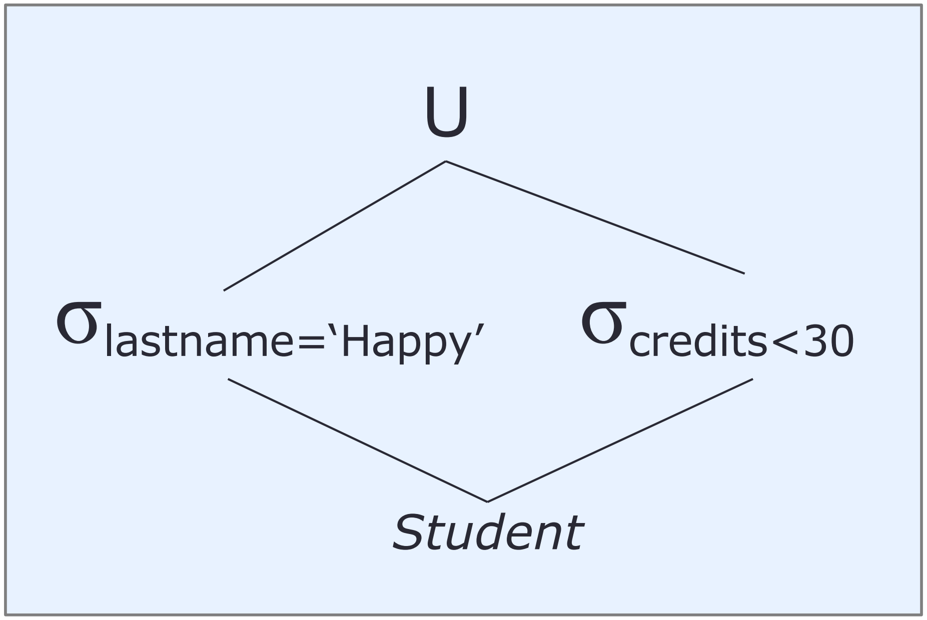
Student

How many tuples do we expect this query to output?
(assume constants exist, distinct values uniformly distributed and continuous; without assumption, estimate is impossible)

Estimation: OR (2)

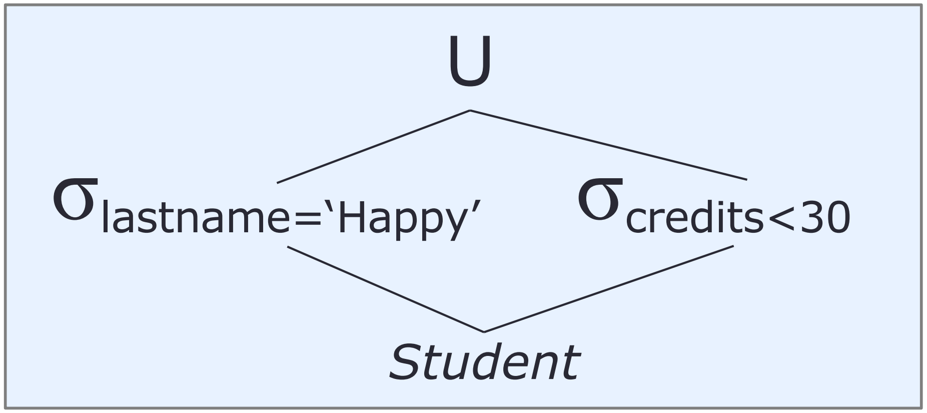
```
SELECT *
FROM Student
WHERE credits < 30
OR lastname = 'Happy'
```





Estimation: OR (3)

```
SELECT *
FROM Student
WHERE credits < 30
OR lastname = ‘Happy’
```


U
$\sigma_{\text{lastname='Happy'}}$
$\sigma_{\text{credits<30}}$
Student

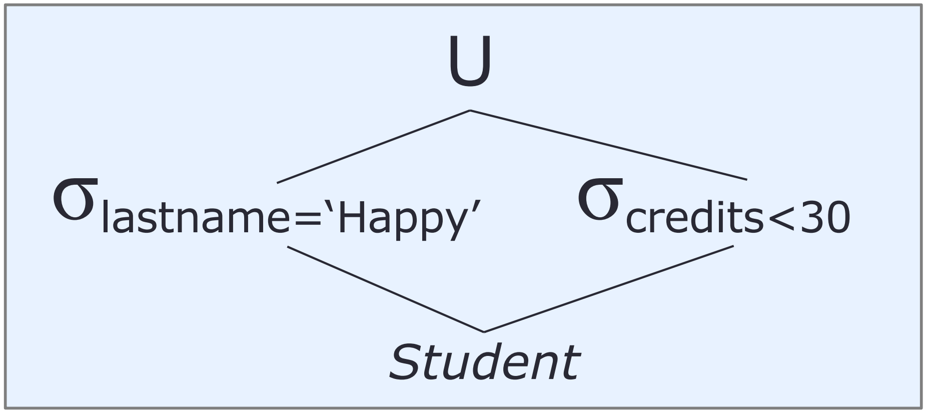
How many tuples do we expect this query to output? – hard to say

- If conditions disjoint, add estimates
  - E.g. no student with lastname ‘Happy’ has credits <30
- If conditions fully overlap, take maximum of estimates
  - E.g. all students with lastname ‘Happy’ have credits <30

Assume disjoint unless you know for sure full overlap

Estimation: OR (4)

```
SELECT *
FROM Student
WHERE credits < 30
OR lastname = 'Happy'
```



$$\cup$$

$\sigma_{lastname=\text{'Happy'}}$ $\sigma_{credits<30}$

Student

How many tuples do we expect this query to output? – hard to say

- If conditions disjoint, add estimates

$\leq 10000 * \boxed{((30-1) / (126-1))} + (10000 * \boxed{1/9500}) \approx 2321$ tuples

- If conditions fully overlap, take maximum of estimates

$\geq 10000 * \max\{ \boxed{((30-1) / (126-1))}, (\boxed{1/9500}) \} \approx 2320$ tuples

(assume disjoint unless otherwise specified --- answer: 2321 tuples)

Selectivity factor

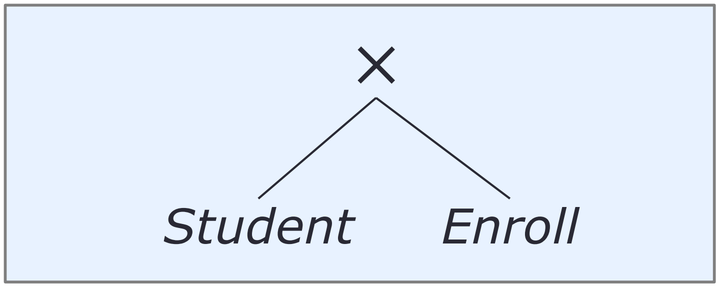
Estimation: Cartesian Product

Assume we know the following information:

```
Student (studId, lastName, fistName, major, credits)
  T(Student) = 10,000                    #of tuples
  V(lastName) = 9,500                    #of distinct values
  V(major) = 10                          #of distinct values
  Range(credits) = [1, 126)              range of values

Enroll (studId, classNumber, grade)
  T(Enroll) = 50,000                     #of tuples
  V(studId) = 10,000                     #of distinct values
  V(classNumber) = 200                   #of distinct values
```

```sql
SELECT *
FROM Student, Enroll
```



How many tuples do we expect this query to output?

T(Student) * T(Enroll) = 10000 * 50000 tuples

No selectivity factor (because no WHERE clause applied)

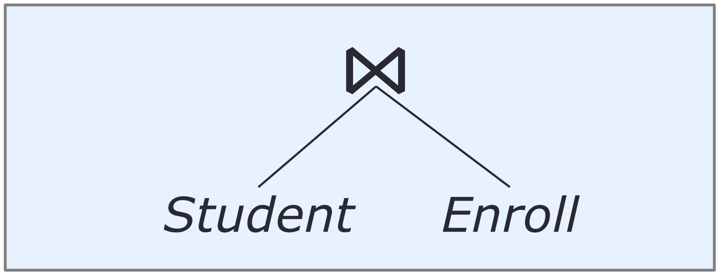
Estimation: JOIN

Assume we know the following information:

```
Student (studId, lastName, fistName, major, credits)
  T(Student) = 10,000                  #of tuples
  V(lastName) = 9,500                  #of distinct values
  V(major) = 10                        #of distinct values
  Range(credits) = [1, 126)            range of values

Enroll (studId, classNumber, grade)
  T(Enroll) = 50,000                   #of tuples
  V(studId) = 10,000                   #of distinct values
  V(classNumber) = 200                 #of distinct values
```

```
SELECT *
FROM Student
NATURAL JOIN Enroll
```



How many tuples do we expect this query to output?

≤ T(Student) * T(Enroll)

≤ 10000 * 50000 tuples

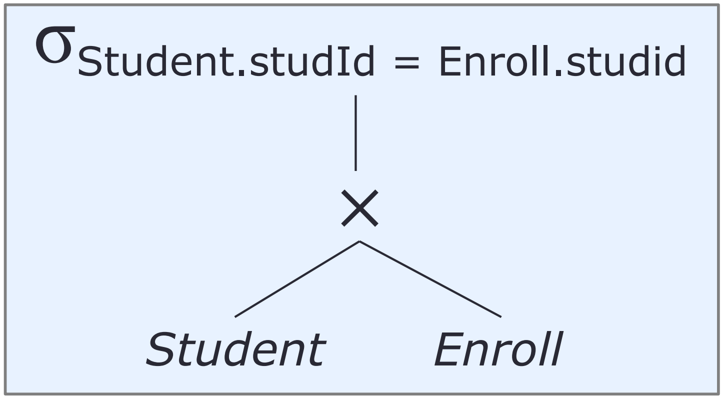
Can we do better?

Estimation: JOIN (2)

1. Start with Cartesian product

```
SELECT *
FROM Student, Enroll
WHERE Student.studId = Enroll.studId
```

T(Student) * T(Enroll)



2. Suppose there are `studId` in both relations that match

3. How many times does `sid0` occur? (assume `sid0` is studentId)

   How many tuples do we expect from

   $\sigma_{\text{Student.studId=sid0 AND Enroll.studId=sid0}}$

Selectivity factor = $\dfrac{1}{V(Student, studId)} * \dfrac{1}{V(Enroll, studId)}$

Estimation: JOIN (3)

4. How many distinct values of `sid0s` exist in the join?

   - If no overlap

     0

   - If full overlap

     ≤ min{ V(Student, studId), V(Enroll, studId) }

Assume full overlap
(~ one is a subset of the other)

5. Multiply (1), (3), and (4)

$$\frac{\mathrm{T(Student)} * \mathrm{T(Enroll)}}{V(Student,studId) * V(Enroll,studId)} * \mathrm{min\{V(Student, studId), V(Enroll, studId)\}}$$

Simplify to

$$\frac{\mathrm{T(Student)} * \mathrm{T(Enroll)}}{\mathrm{max\{V(Student, studId), V(Enroll, studId)\}}}$$

Estimation: JOIN (4)

Assume we know the following information:

Student (studId, lastName, fistName, major, credits)

T(Student) = 10,000	#of tuples
V(lastName) = 9,500	#of distinct values
V(major) = 10	#of distinct values
Range(credits) = [1, 126)	range of values

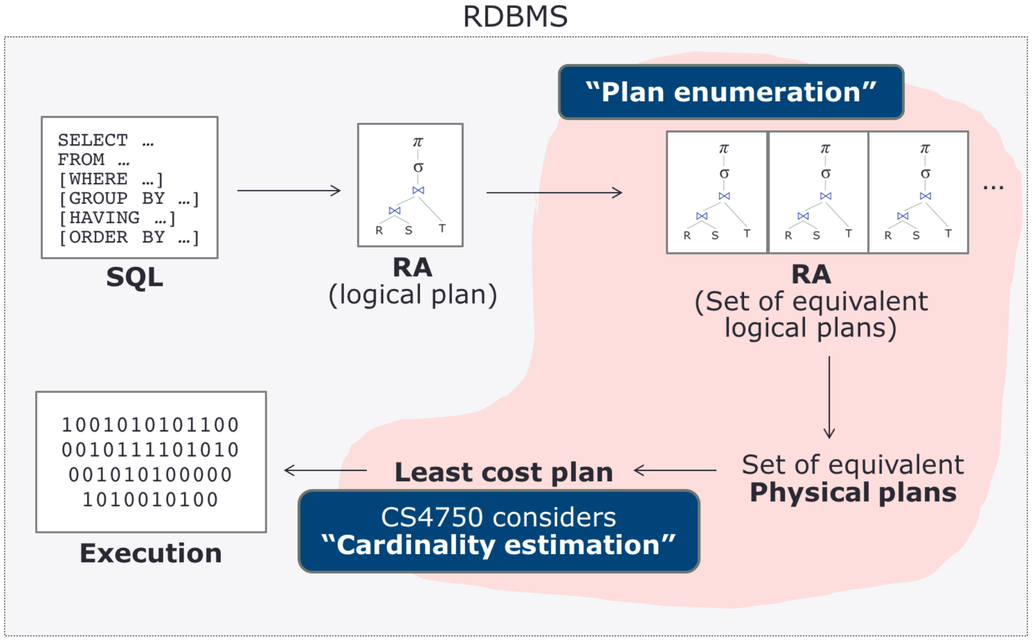
Enroll (studId, classNumber, grade)

T(Enroll) = 50,000	#of tuples
V(studId) = 10,000	#of distinct values
V(classNumber) = 200	#of distinct values

$$\frac{\mathrm{T(Student)} * \mathrm{T(Enroll)}}{\max\{\mathrm{V(Student, studId), V(Enroll, studId)}\}} = \frac{10000 * 50000}{\max\{10000, 10000\}} = 50000 \text{ tuples}$$

Since we assume full overlap of studIds between Student and Enroll, we only need the studIds of the smaller relation

Wrap-Up



- Cardinality estimation
- Real RDBMS uses sophisticated cost model
- Making inappropriate assumptions to estimate cardinality may lead to:
  - Inaccurate estimates
  - Optimization selects a slow plan
  - Slow query execution
- Be careful and document your assumptions

What's next?

- Indexing