CSC3170
Tutorial 11

School of Data Science

The Chinese University of Hong Kong, Shenzhen

Overview

Query Optimization

- Heuristics
  - Relational Algebra Equivalences
  - Logical Query Optimization
  - Nested Queries
  - Expression Rewriting
- Exercise

Why we need Query Optimization

Remember that SQL is declarative.
--User tells the DBMS what answer they want, not how to get the answer.

There can be a big difference in performance based on plan is used

Heuristics / Rules

--Rewrite the query to remove stupid / inefficient things.

--These techniques may need to examine catalog, but they do not need to examine data.

Cost-based Search

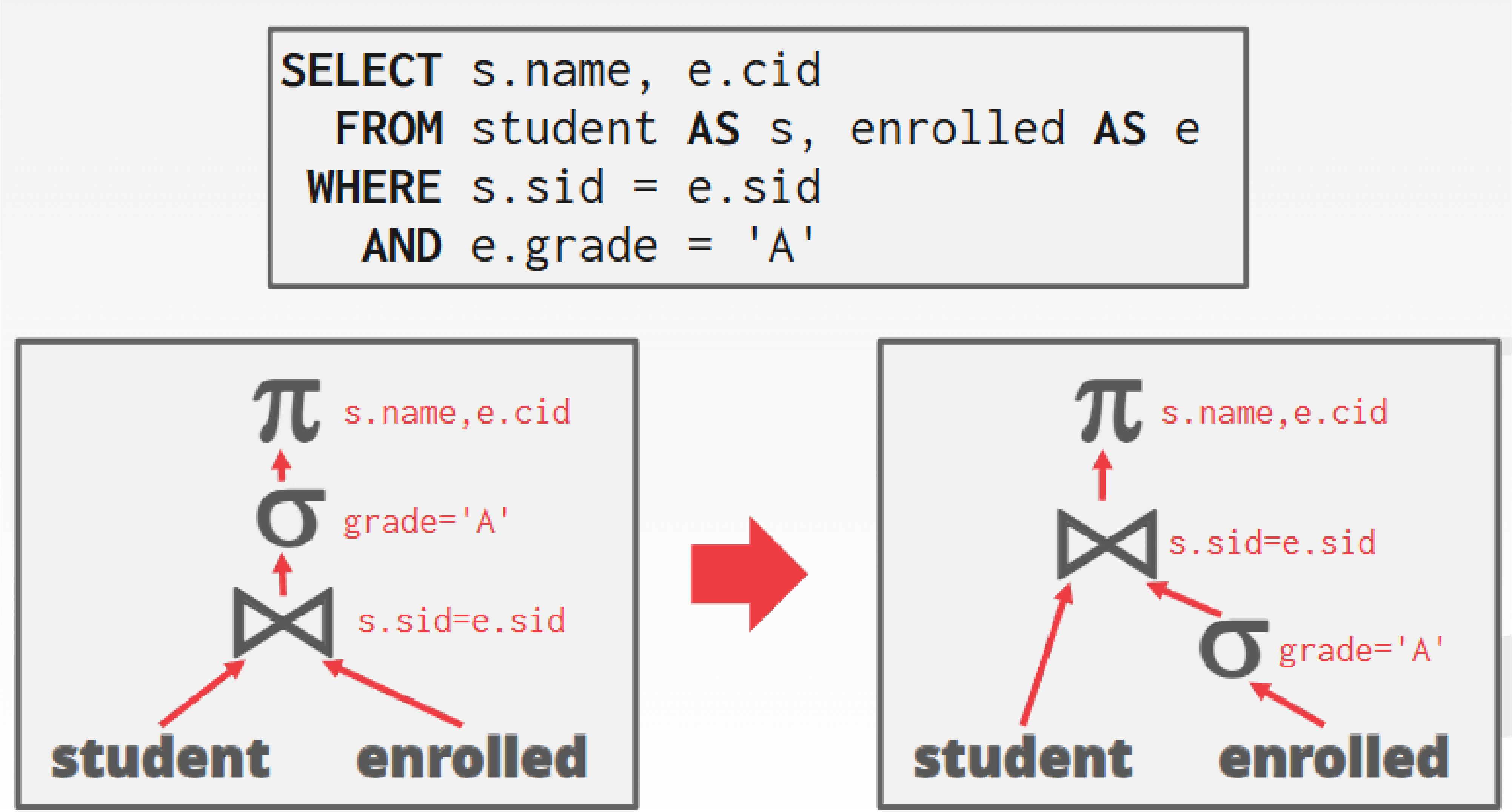
--Use a model to estimate the cost of executing a plan.

--Evaluate multiple equivalent plans for a query and pick the one with the lowest cost.

A DBMS can use **Heuristics/Rules** to transform relational algebra expressions into equivalent expressions with lower costs, thereby achieving query optimization. These rules are typically applied to all queries. For example:

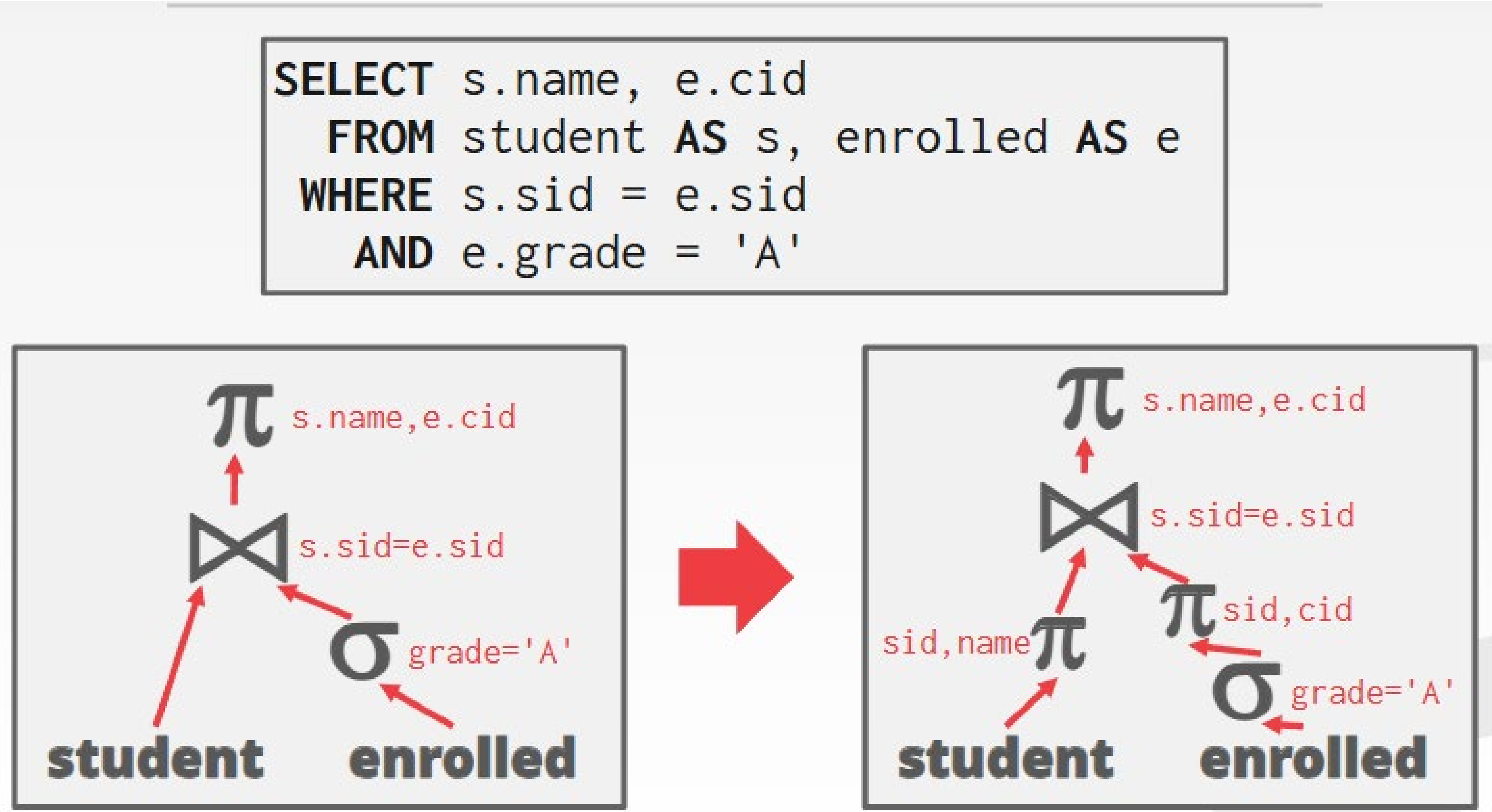
Predicate Pushdown
Projections Pushdown
Nested Query Rewriting
Expression Rewriting
……

Predicate Pushdown

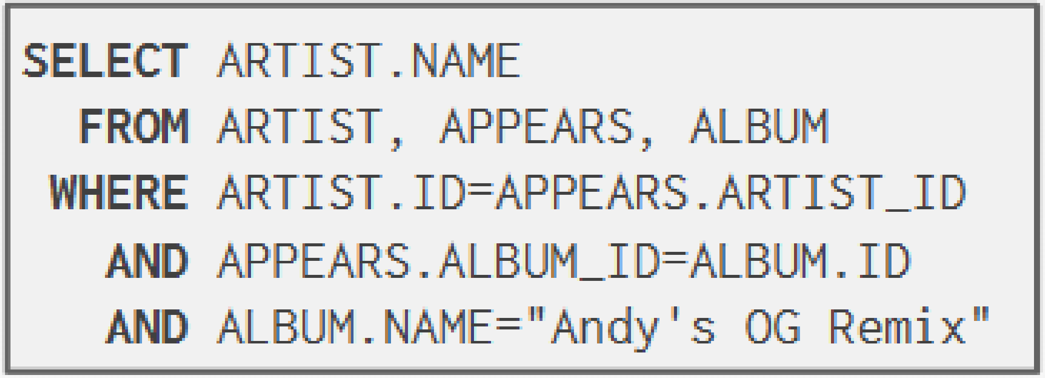


```
SELECT s.name, e.cid
  FROM student AS s, enrolled AS e
 WHERE s.sid = e.sid
   AND e.grade = 'A'
```

Projections Pushdown



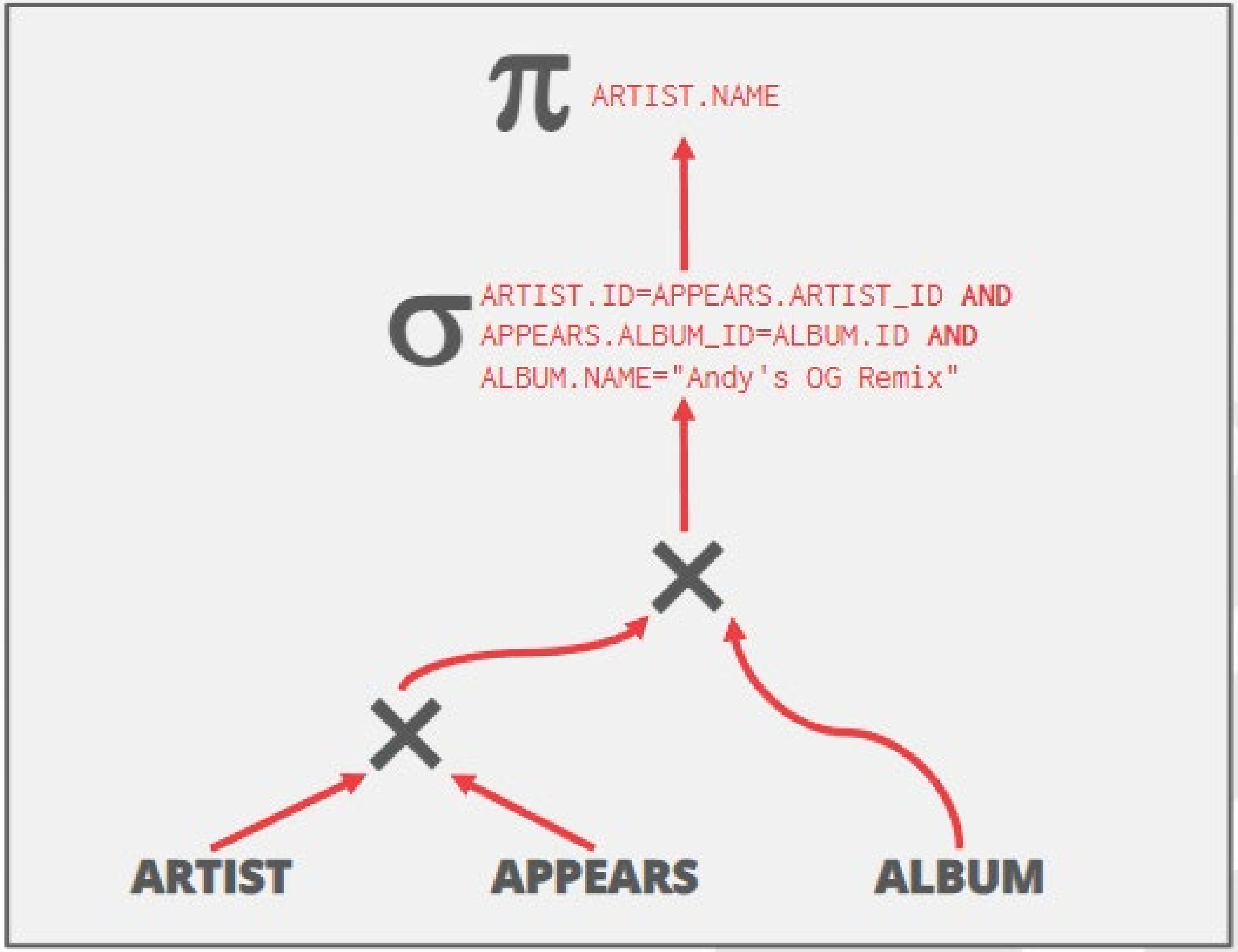
```sql
SELECT s.name, e.cid
  FROM student AS s, enrolled AS e
 WHERE s.sid = e.sid
   AND e.grade = 'A'
```

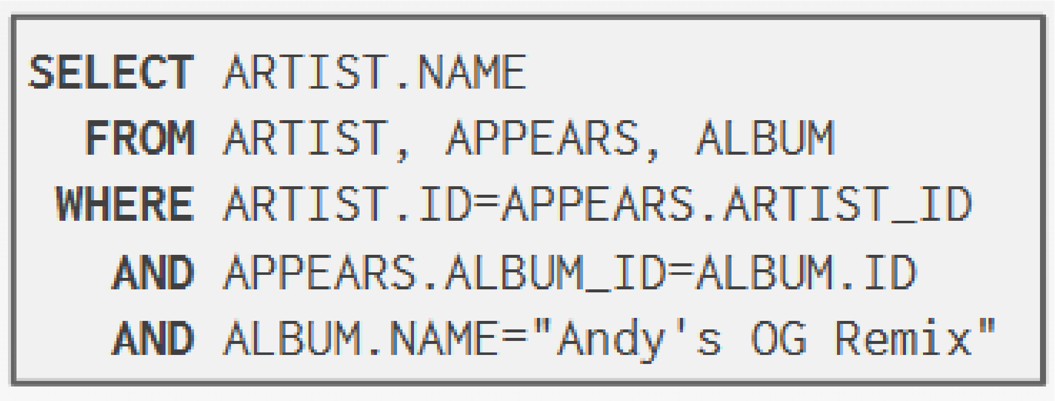


```
SELECT ARTIST.NAME
  FROM ARTIST, APPEARS, ALBUM
 WHERE ARTIST.ID=APPEARS.ARTIST_ID
   AND APPEARS.ALBUM_ID=ALBUM.ID
   AND ALBUM.NAME="Andy's OG Remix"
```

Decompose predicates into their simplest forms

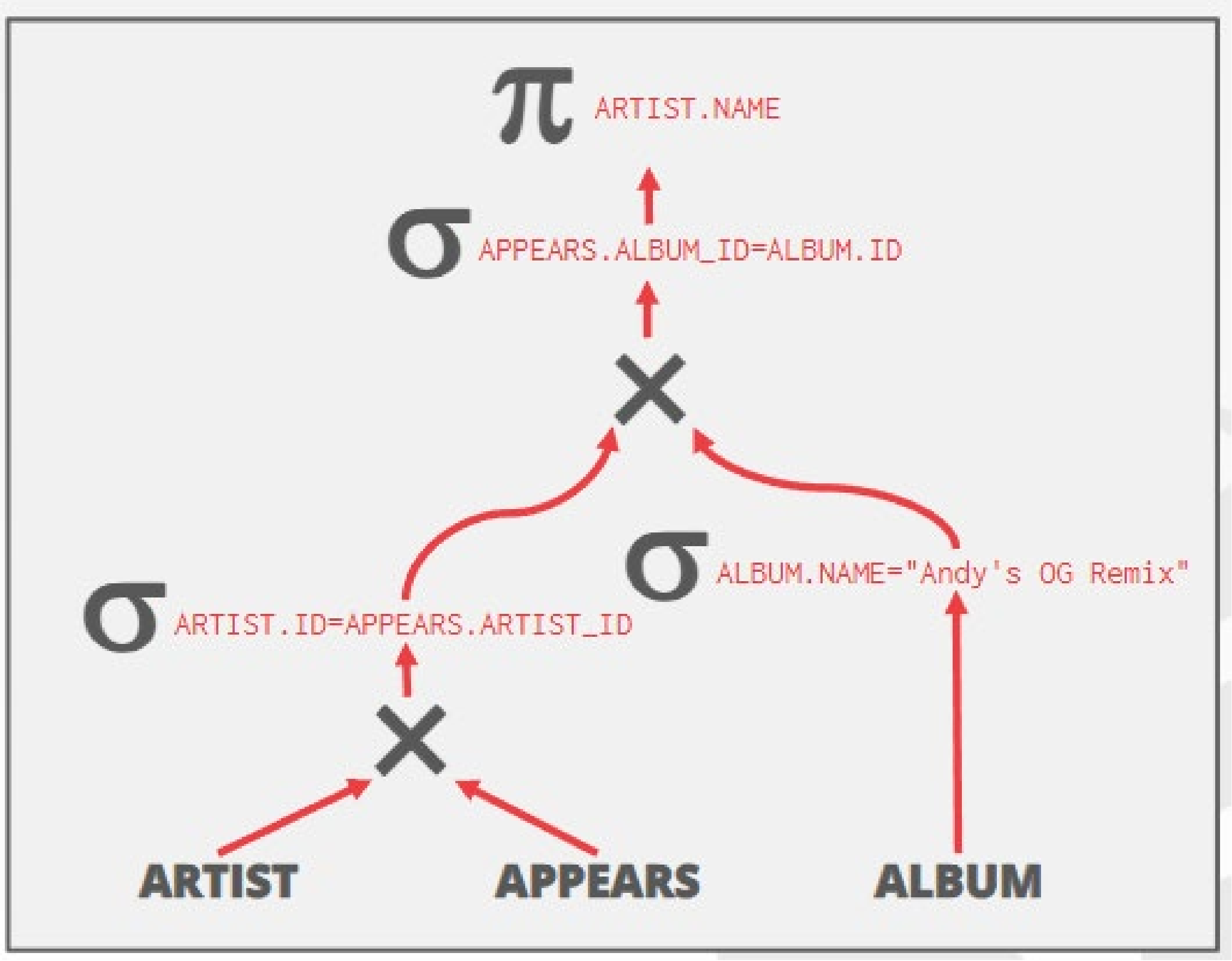
Move the predicate to the lowest applicable point in the plan.

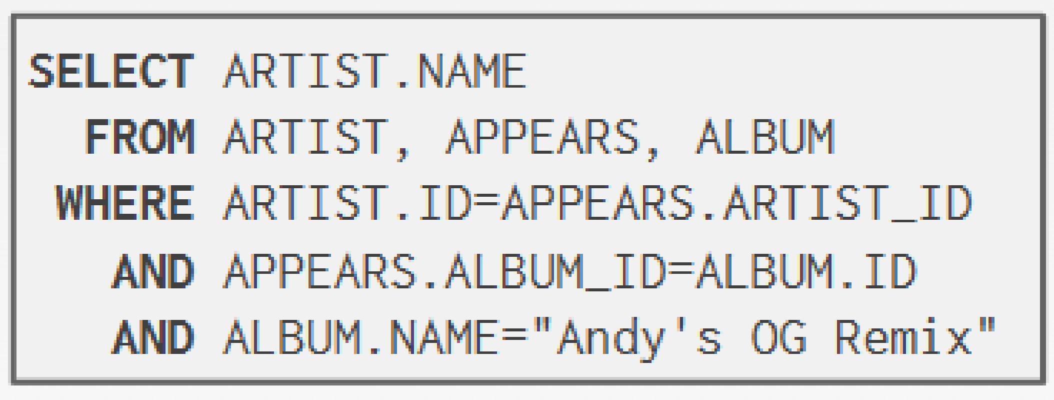

π ARTIST.NAME
σ ARTIST.ID=APPEARS.ARTIST_ID AND APPEARS.ALBUM_ID=ALBUM.ID AND ALBUM.NAME="Andy's OG Remix"
×
×
ARTIST
APPEARS
ALBUM



```
SELECT ARTIST.NAME
  FROM ARTIST, APPEARS, ALBUM
 WHERE ARTIST.ID=APPEARS.ARTIST_ID
   AND APPEARS.ALBUM_ID=ALBUM.ID
   AND ALBUM.NAME="Andy's OG Remix"
```

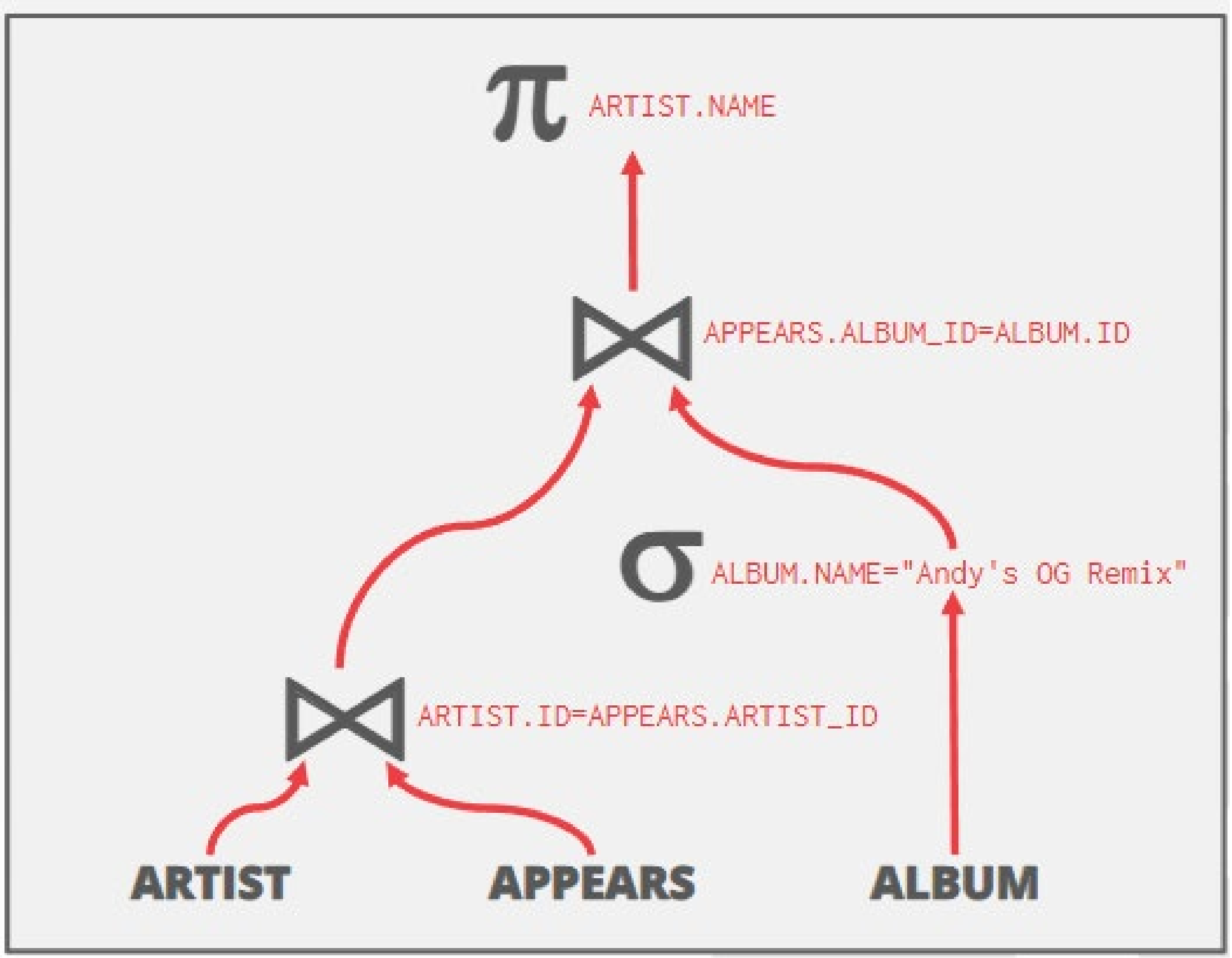
Replace all Cartesian Products with inner joins using the join predicates.

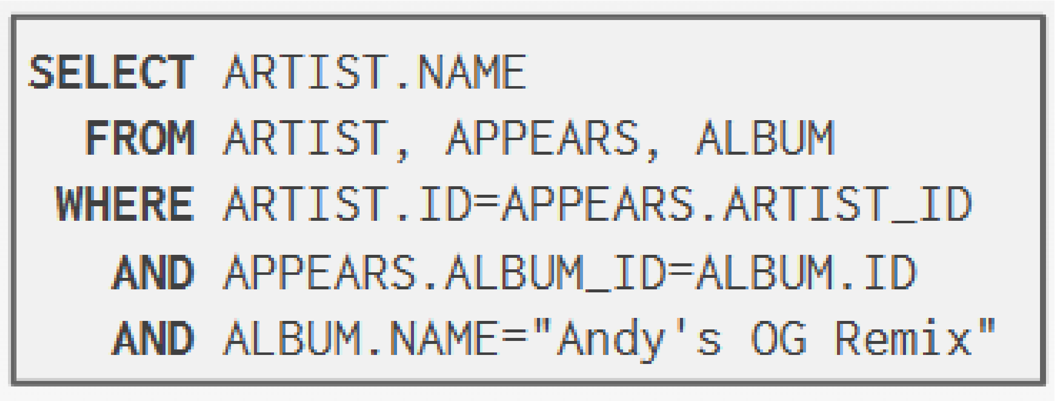

π ARTIST.NAME
σ APPEARS.ALBUM_ID=ALBUM.ID
×
σ ARTIST.ID=APPEARS.ARTIST_ID
σ ALBUM.NAME="Andy's OG Remix"
×
ARTIST
APPEARS
ALBUM



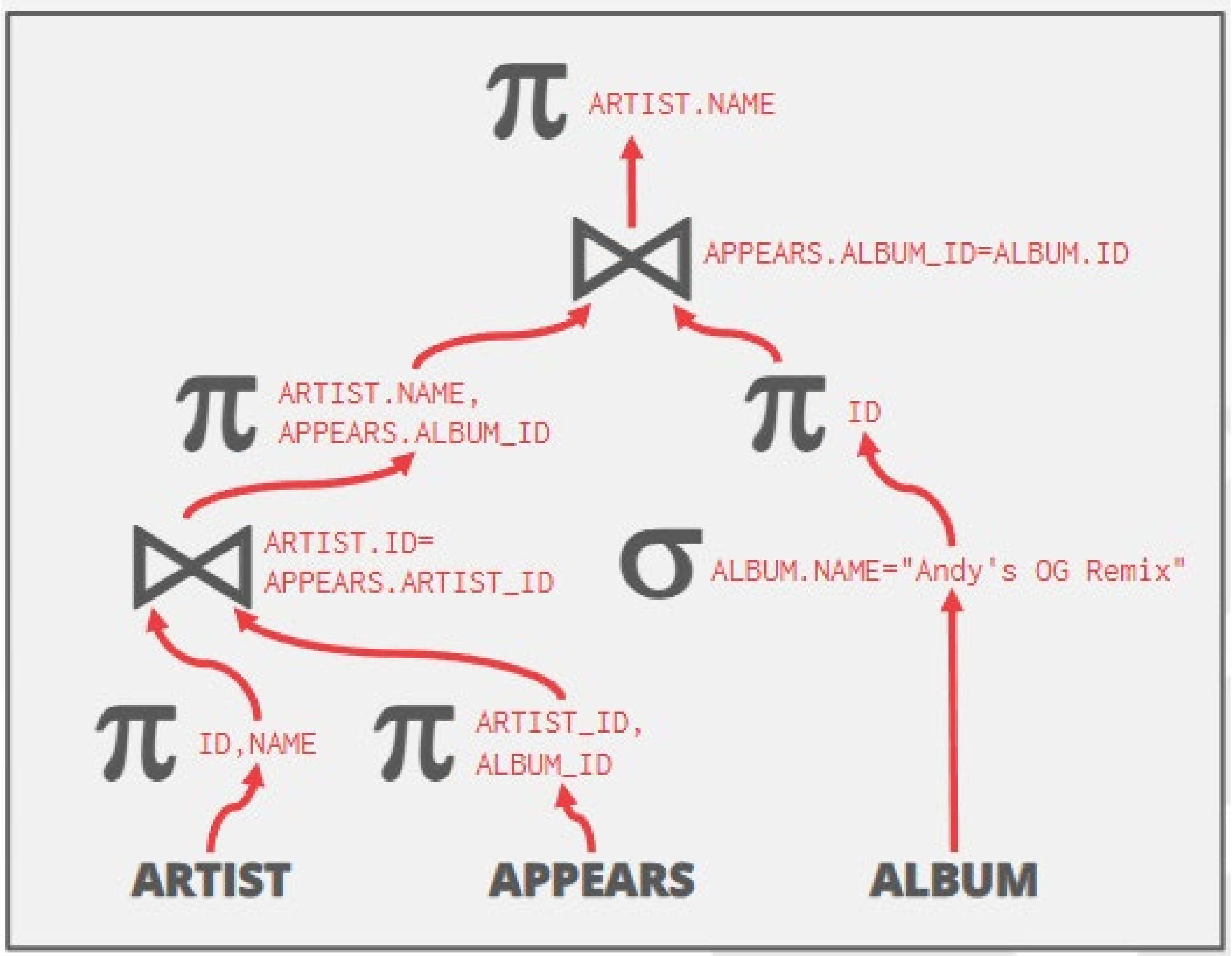
```
SELECT ARTIST.NAME
  FROM ARTIST, APPEARS, ALBUM
 WHERE ARTIST.ID=APPEARS.ARTIST_ID
   AND APPEARS.ALBUM_ID=ALBUM.ID
   AND ALBUM.NAME="Andy's OG Remix"
```

Eliminate redundant attributes before pipeline breakers to reduce materialization cost.


π ARTIST.NAME
⋈ APPEARS.ALBUM_ID=ALBUM.ID
σ ALBUM.NAME="Andy's OG Remix"
⋈ ARTIST.ID=APPEARS.ARTIST_ID
ARTIST
APPEARS
ALBUM

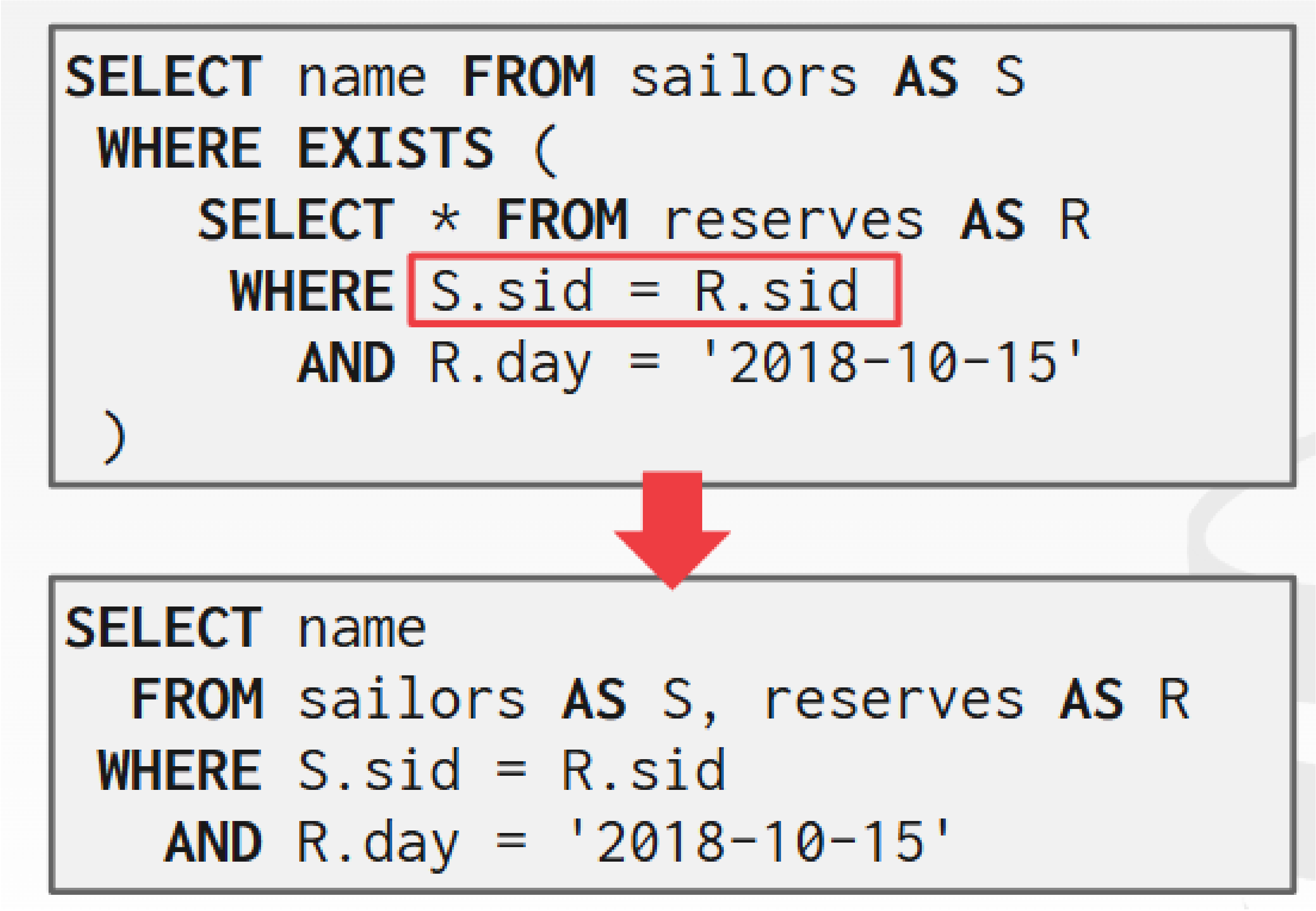


```
SELECT ARTIST.NAME
  FROM ARTIST, APPEARS, ALBUM
 WHERE ARTIST.ID=APPEARS.ARTIST_ID
   AND APPEARS.ALBUM_ID=ALBUM.ID
   AND ALBUM.NAME="Andy's OG Remix"
```


π ARTIST.NAME
⋈ APPEARS.ALBUM_ID=ALBUM.ID
π ARTIST.NAME, APPEARS.ALBUM_ID
π ID
⋈ ARTIST.ID= APPEARS.ARTIST_ID
σ ALBUM.NAME="Andy's OG Remix"
π ID,NAME
π ARTIST_ID, ALBUM_ID
ARTIST
APPEARS
ALBUM

Eliminate redundant attributes before pipeline breakers to reduce materialization cost.

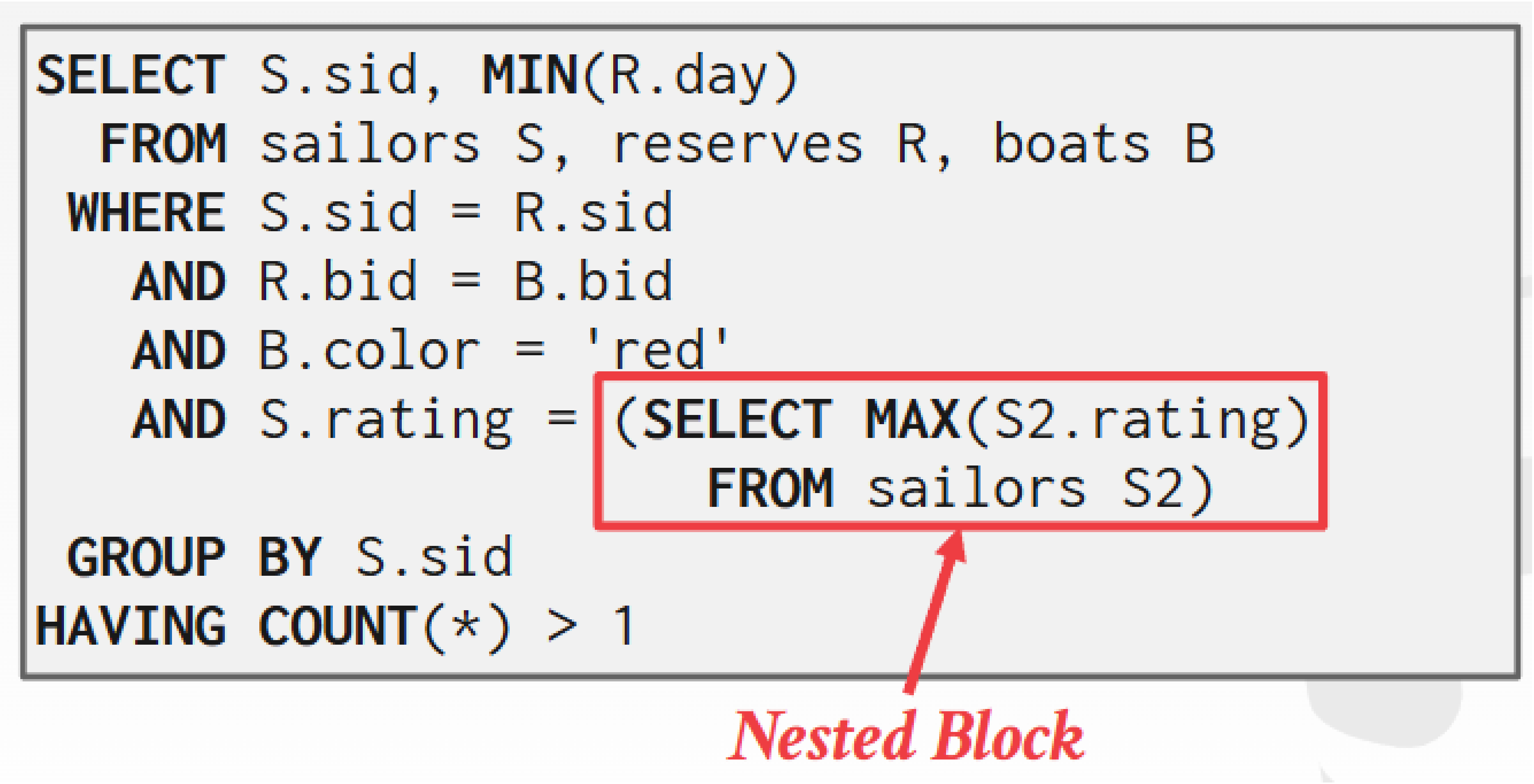
Nested Query Rewrite



```
SELECT name FROM sailors AS S
 WHERE EXISTS (
    SELECT * FROM reserves AS R
     WHERE S.sid = R.sid
       AND R.day = '2018-10-15'
 )
```

```
SELECT name
  FROM sailors AS S, reserves AS R
 WHERE S.sid = R.sid
   AND R.day = '2018-10-15'
```

Nested Query Decompose

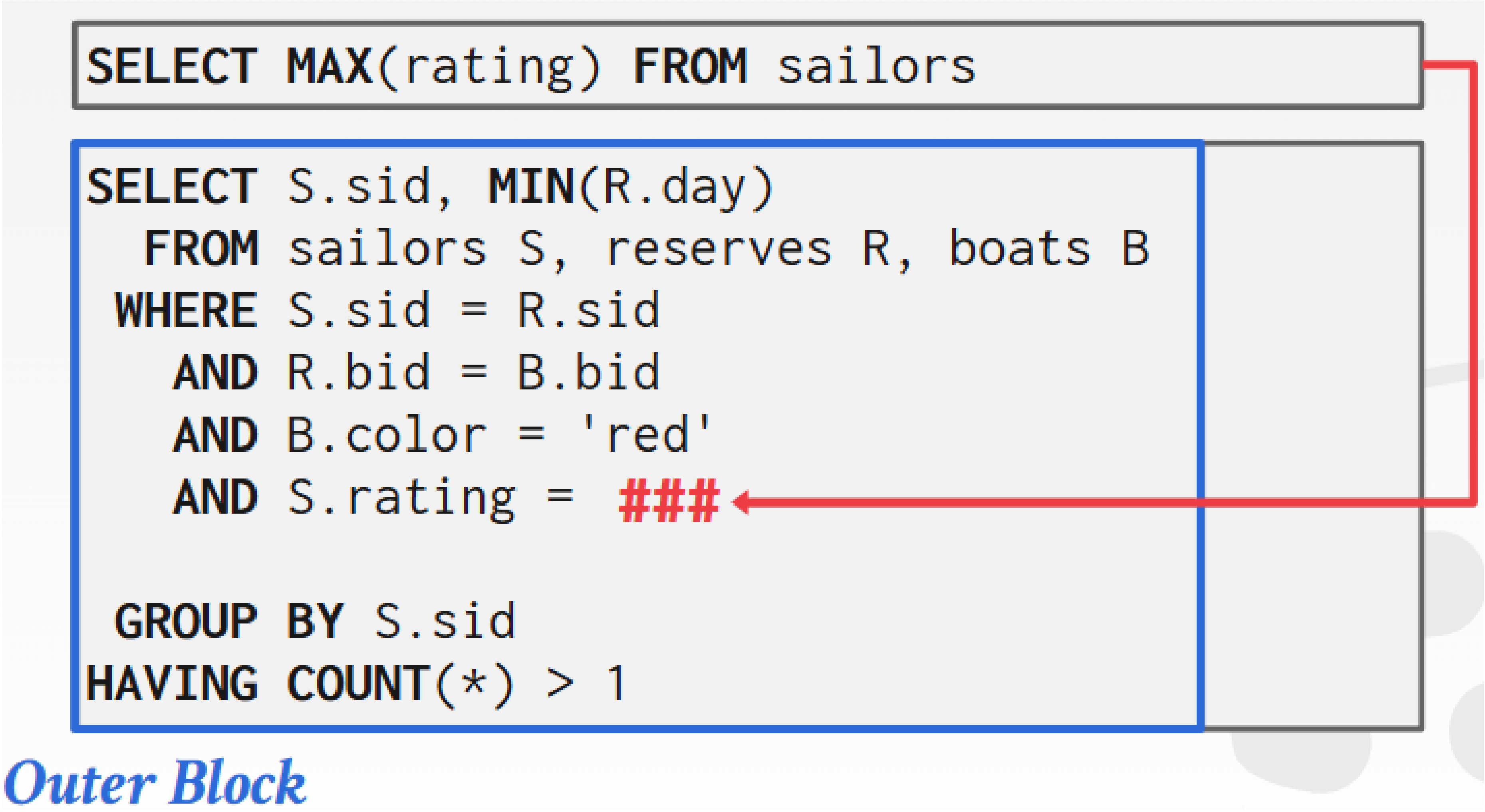


```
SELECT S.sid, MIN(R.day)
  FROM sailors S, reserves R, boats B
 WHERE S.sid = R.sid
   AND R.bid = B.bid
   AND B.color = 'red'
   AND S.rating = (SELECT MAX(S2.rating)
                     FROM sailors S2)

 GROUP BY S.sid
HAVING COUNT(*) > 1
```

Nested Block

Nested Query Decompose



```
SELECT MAX(rating) FROM sailors
```

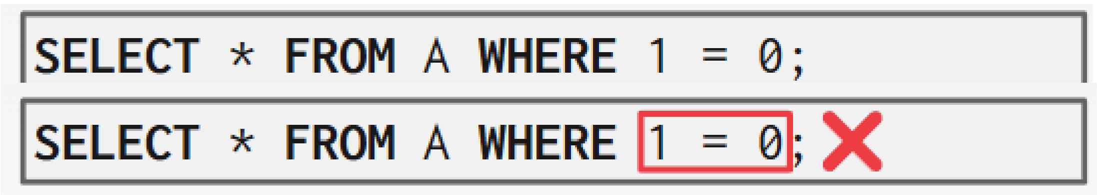
```
SELECT S.sid, MIN(R.day)
  FROM sailors S, reserves R, boats B
 WHERE S.sid = R.sid
   AND R.bid = B.bid
   AND B.color = 'red'
   AND S.rating = ###

 GROUP BY S.sid
HAVING COUNT(*) > 1
```

Outer Block

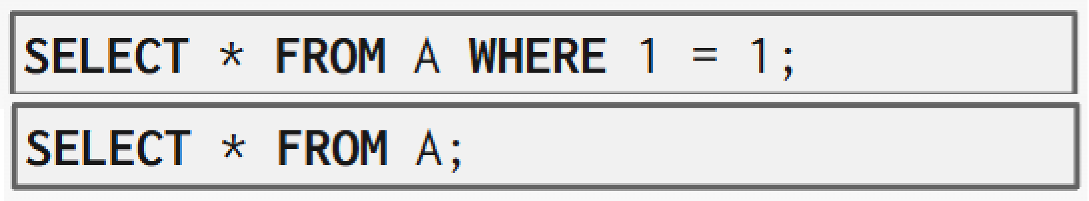
Expression Rewrite

Impossible/Unnecessary predicate



```
SELECT * FROM A WHERE 1 = 0;
```

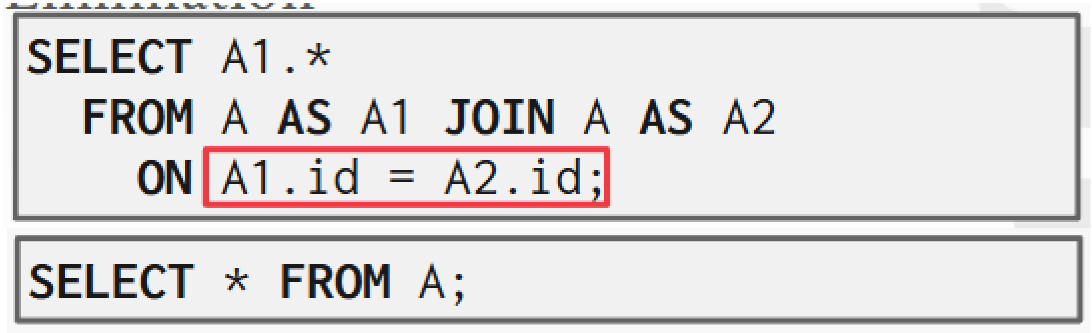
```
SELECT * FROM A WHERE 1 = 0; ✗
```



```
SELECT * FROM A WHERE 1 = 1;
```

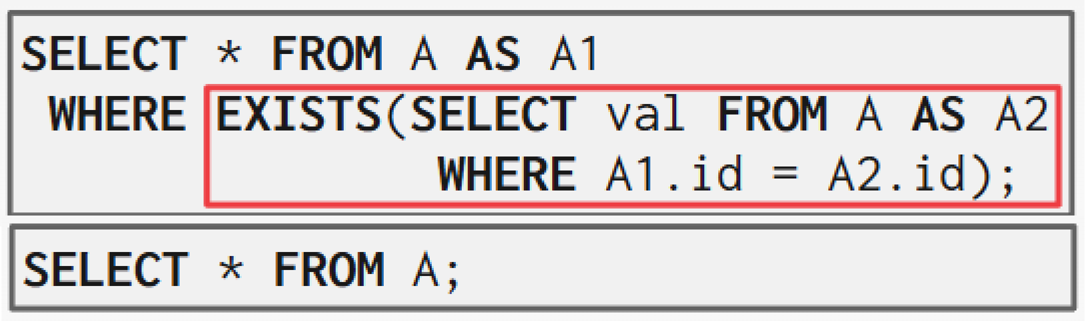
```
SELECT * FROM A;
```

Join estimate



```
SELECT A1.*
  FROM A AS A1 JOIN A AS A2
    ON A1.id = A2.id;
```

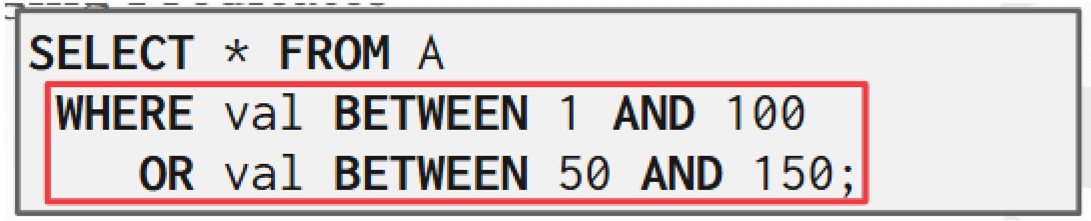
```
SELECT * FROM A;
```



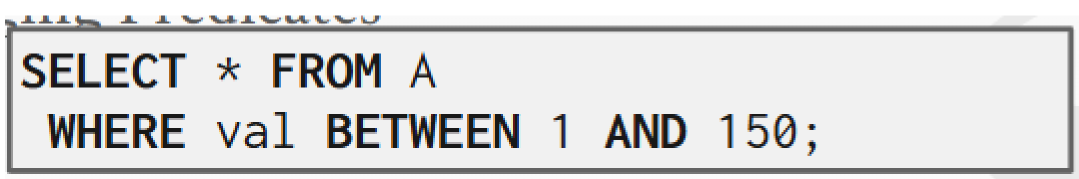
```
SELECT * FROM A AS A1
 WHERE EXISTS(SELECT val FROM A AS A2
                WHERE A1.id = A2.id);
```

```
SELECT * FROM A;
```

Predicates merge



```
SELECT * FROM A
 WHERE val BETWEEN 1 AND 100
    OR val BETWEEN 50 AND 150;
```



```
SELECT * FROM A
 WHERE val BETWEEN 1 AND 150;
```

Exercise

For the student-course database, query the names of all courses taken by students in the Faculty of Computer Science:

```
Select Cname
From Student, Course, Score
Where Student.Sno=Score.Sno and Score.Cno=Course.Cno and Student.Sdept=“CS"
```

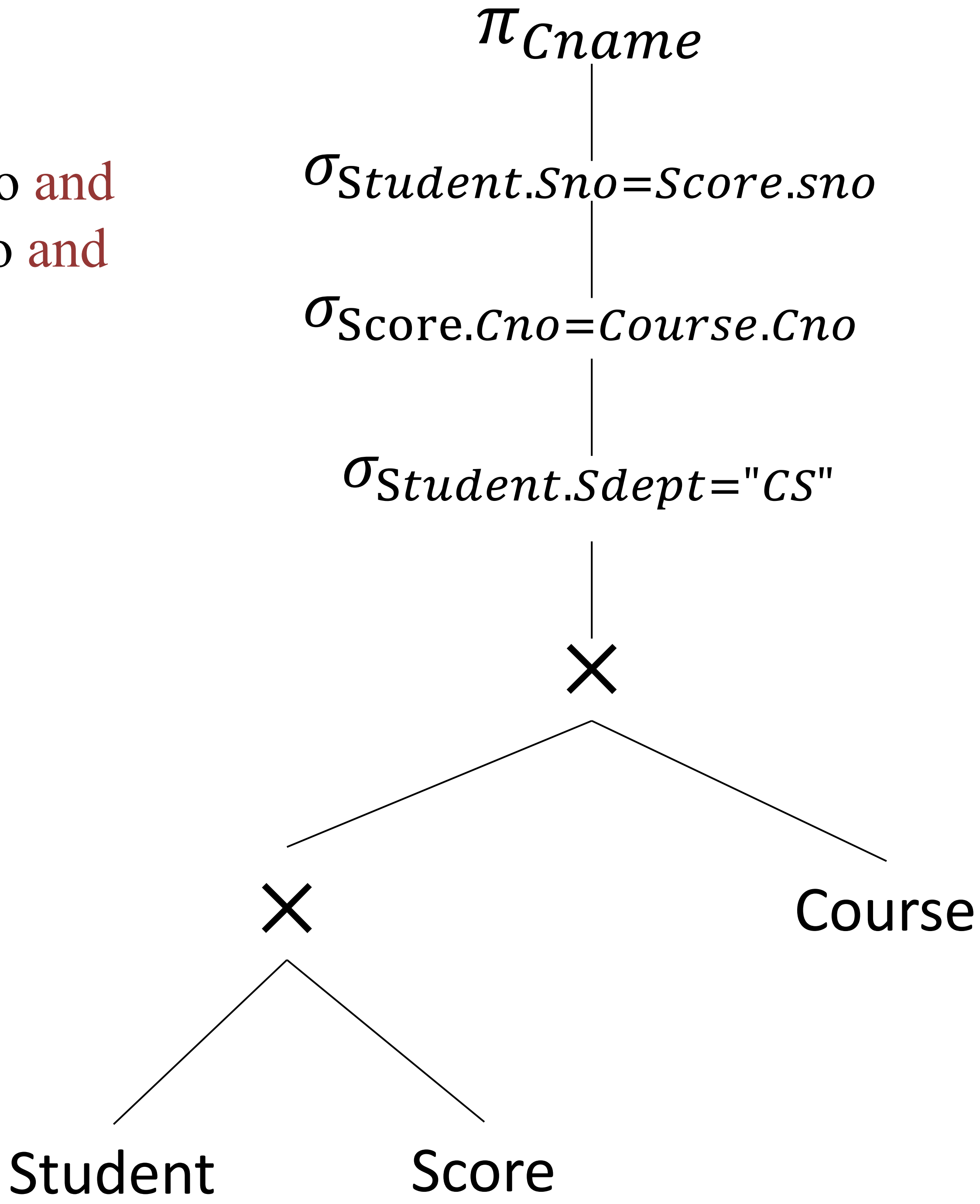
Draw the syntax tree and optimize the syntax tree with relational algebra and draw the optimized syntax tree

Select Cname
From Student, Course, Score
Where Student.Sno=Score.Sno and
Score.Cno=Course.Cno and
Student.Sdept="CS"

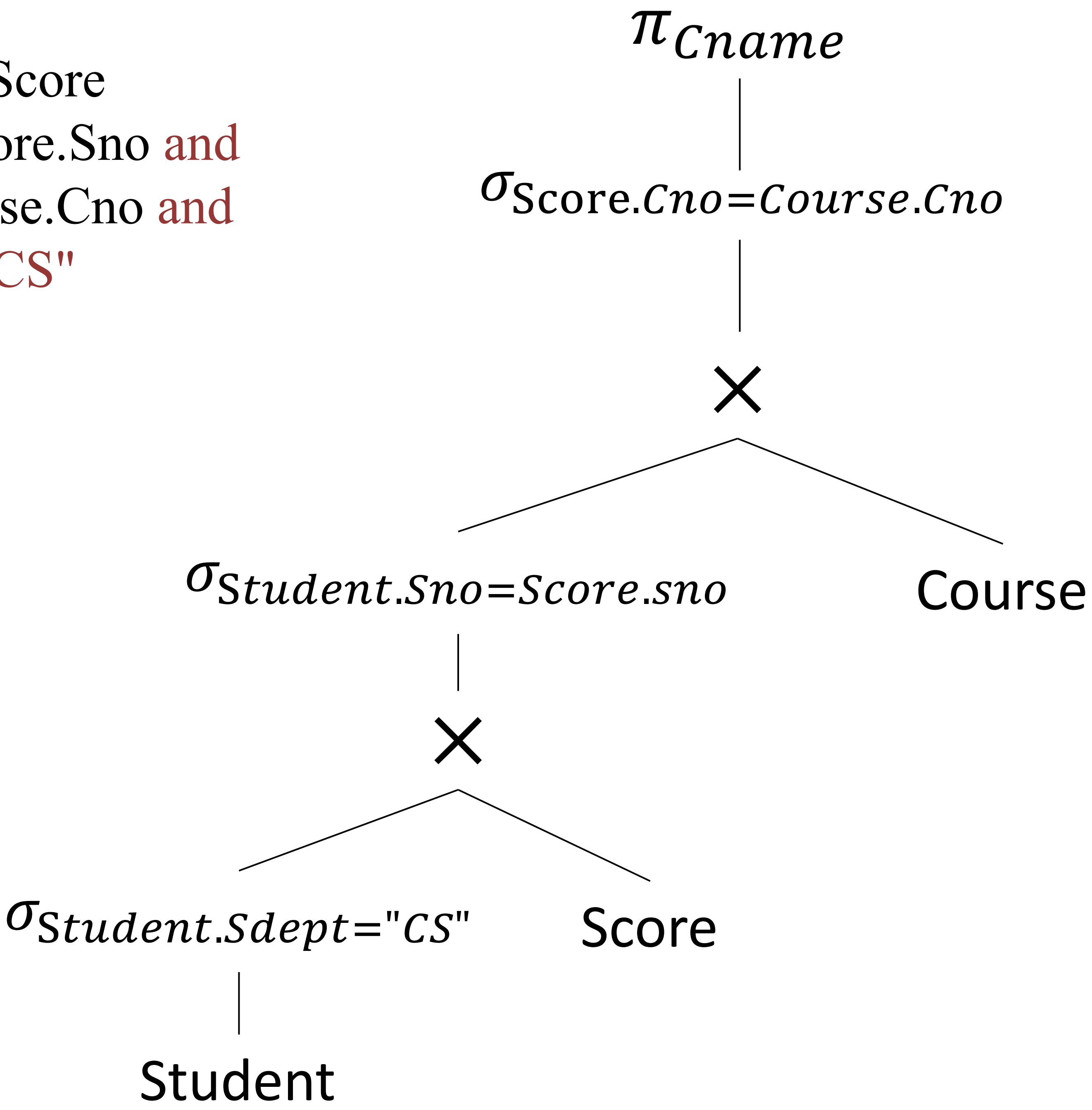
1.Predicate Pushdown.

2.Replacing the Cartesian product using Inner join.

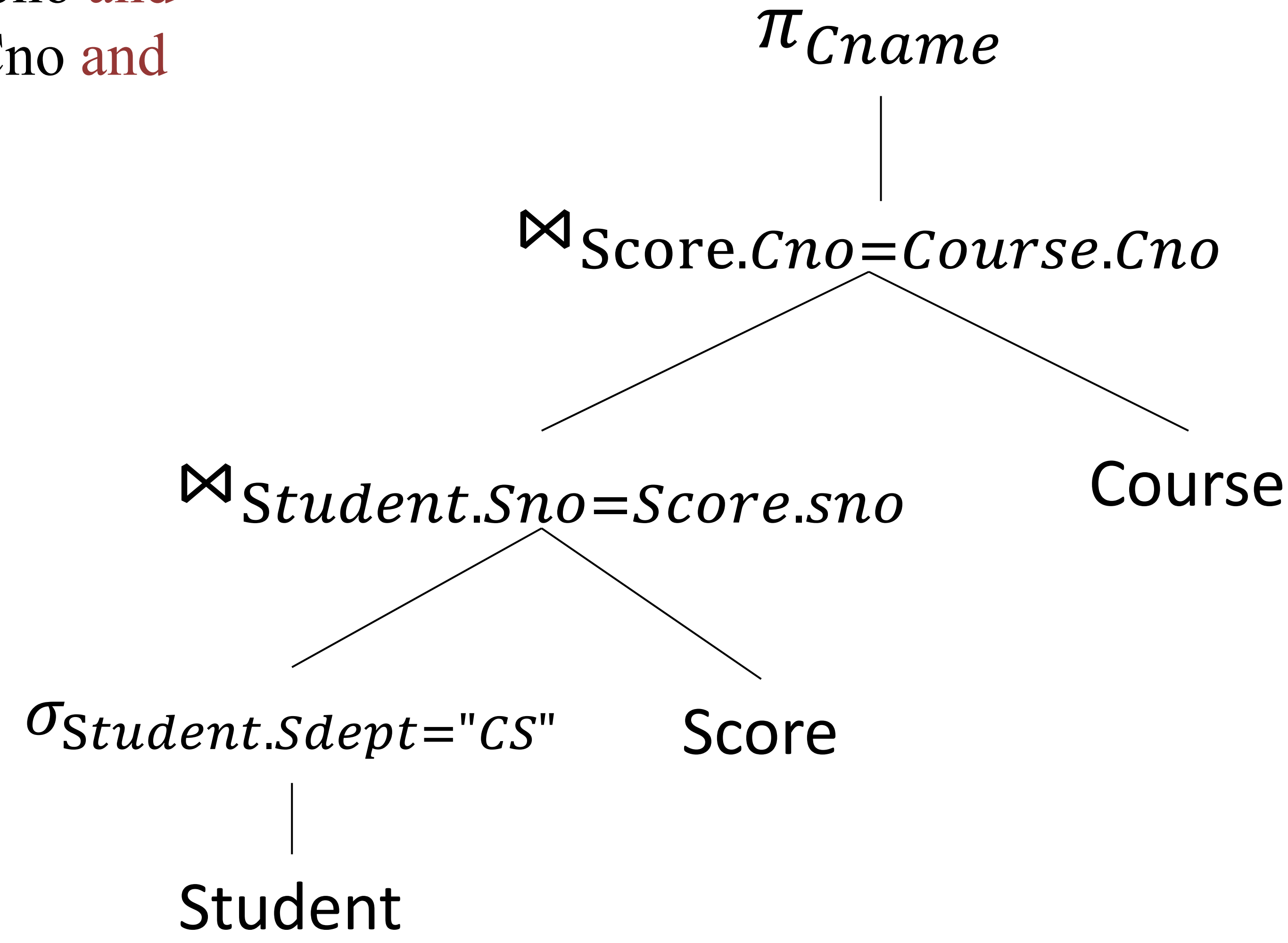
3.Projection Pushdown.


π_{Cname}
$\sigma_{Student.Sno=Score.sno}$
$\sigma_{Score.Cno=Course.Cno}$
$\sigma_{Student.Sdept="CS"}$
×
×
Course
Student
Score

```
Select Cname
From Student, Course, Score
Where Student.Sno=Score.Sno and
         Score.Cno=Course.Cno and
         Student.Sdept=“CS"
```


π_{Cname}
$\sigma_{\mathrm{Score}.Cno=Course.Cno}$
×
$\sigma_{Student.Sno=Score.sno}$
Course
×
$\sigma_{Student.Sdept="CS"}$
Score
Student

```
Select Cname
From Student, Course, Score
Where Student.Sno=Score.Sno and
        Score.Cno=Course.Cno and
        Student.Sdept=“CS"
```



π_{Cname}

$\bowtie_{\text{Score}.Cno=Course.Cno}$

$\bowtie_{Student.Sno=Score.sno}$

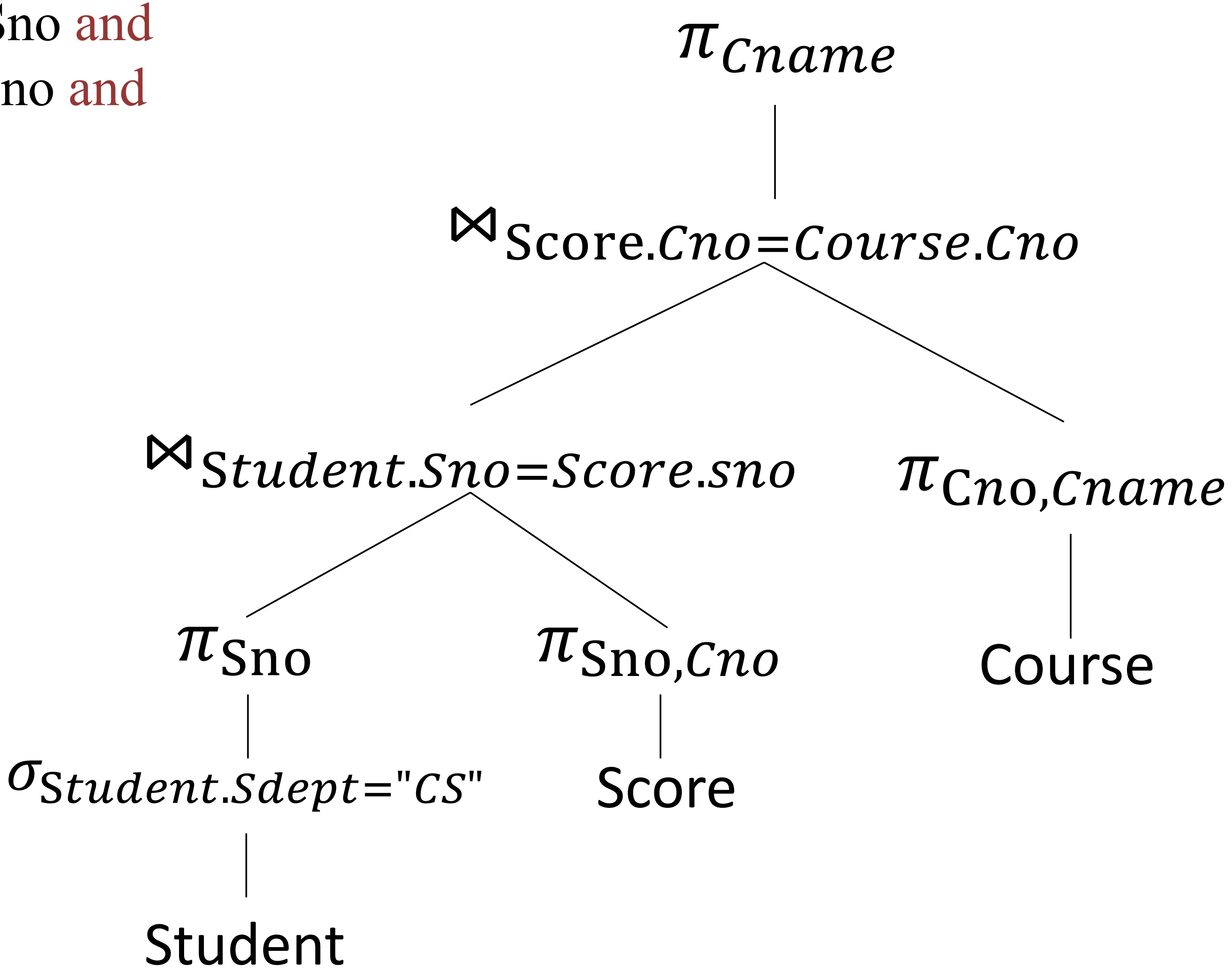
Course

$\sigma_{Student.Sdept="CS"}$

Score

Student

```
Select Cname
From Student, Course, Score
Where Student.Sno=Score.Sno and
        Score.Cno=Course.Cno and
        Student.Sdept=“CS"
```



π_{Cname}

$\bowtie_{Score.Cno=Course.Cno}$

$\bowtie_{Student.Sno=Score.sno}$

$\pi_{Cno,Cname}$

π_{Sno}

$\pi_{Sno,Cno}$

Course

$\sigma_{Student.Sdept="CS"}$

Score

Student

Q&A