CS4224/CS5424 Lecture 3
Distributed Query Processing

Query Processing

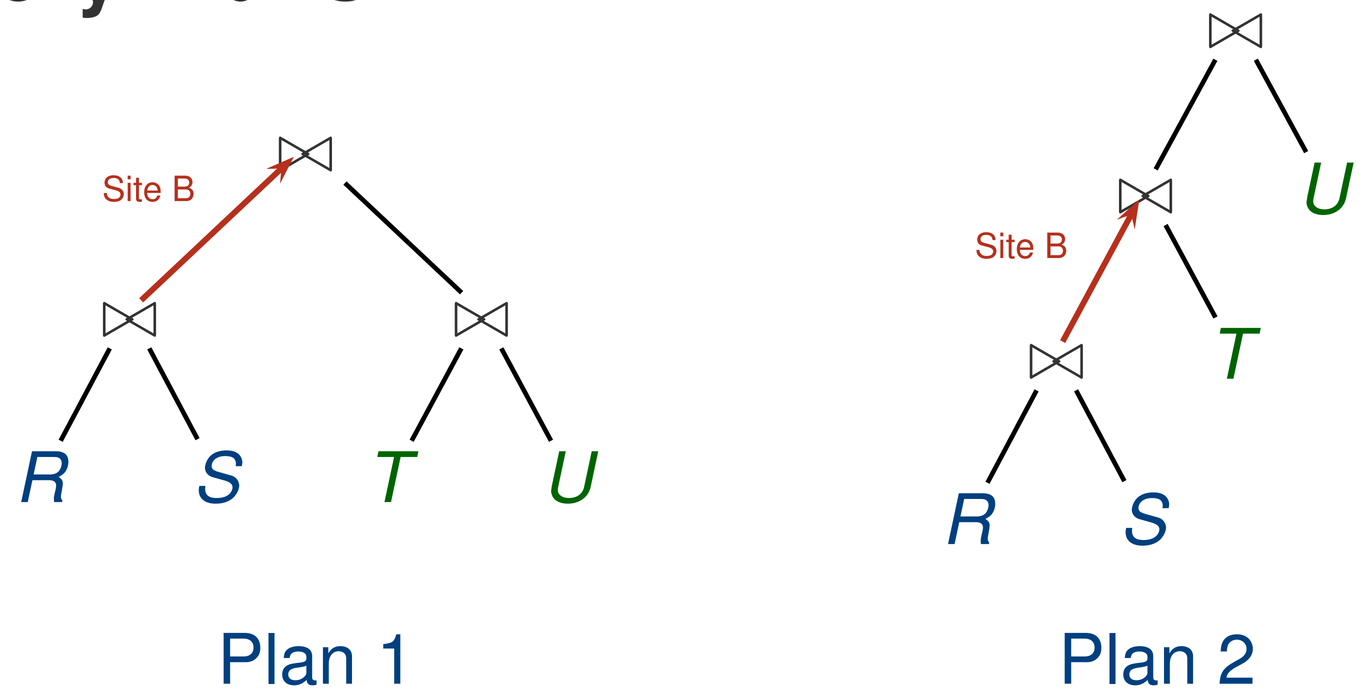
- Translates query into a query plan that minimizes some cost function
  - Minimize total cost
    - CPU cost, I/O cost, & communication cost
  - Minimize response time
    - Time elapsed for query execution

Example

- Site A: Relations $R(a, c, \cdots)$ & $S(a, \cdots)$
- Site B: Relations $T(b, c, \cdots)$ & $U(b, \cdots)$
- **Query at Site B**:

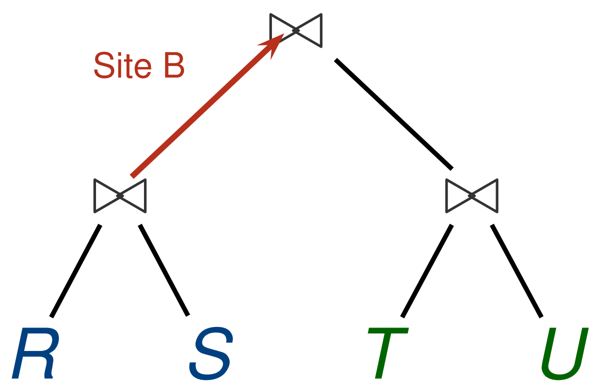
  SELECT * FROM R, S, T, U
  WHERE R.a = S.a AND T.b = U.b AND R.c = T.c

- **Query Plans**:

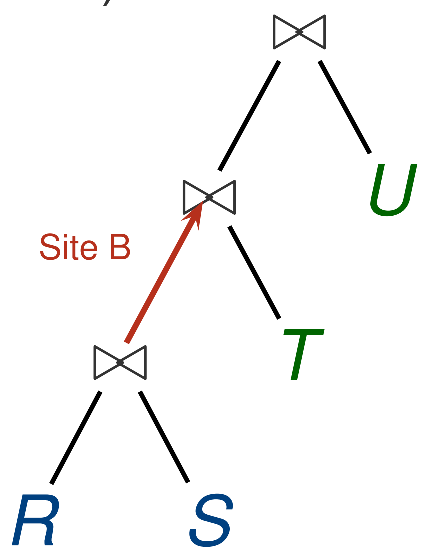


Example (cont.)

- **Cost model:**
  - **JC(X,Y)** = CPU & I/O cost of joining relations X & Y
  - **CC(X)** = Communication cost of sending relation X from one site to another site
  - JC(R, S) = 2000, JC(T, U) = 2000
  - JC($R \bowtie S$, T) = 1000, JC($R \bowtie S \bowtie T$, U) = 600
  - JC($R \bowtie S$, $T \bowtie U$) = 100, CC($R \bowtie S$) = 200



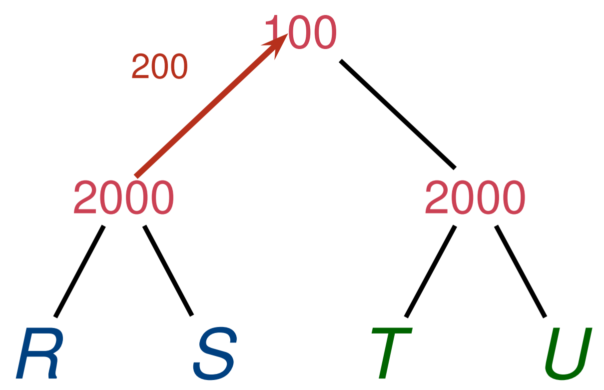
Plan 1

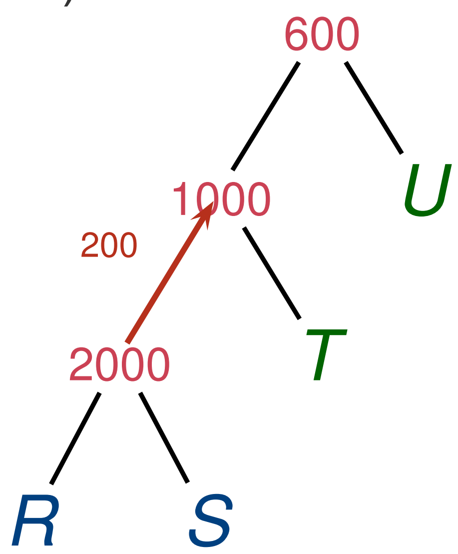


Plan 2

Example (cont.)

- **Cost model:**
  - **JC(X,Y)** = CPU & I/O cost of joining relations X & Y
  - **CC(X)** = Communication cost of sending relation X from one site to another site
  - JC(R, S) = 2000, JC(T, U) = 2000
  - JC($R \bowtie S$, T) = 1000, JC($R \bowtie S \bowtie T$, U) = 600
  - JC($R \bowtie S$, $T \bowtie U$) = 100, CC($R \bowtie S$) = 200



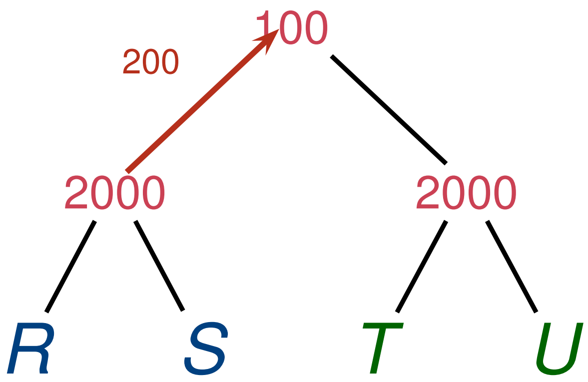


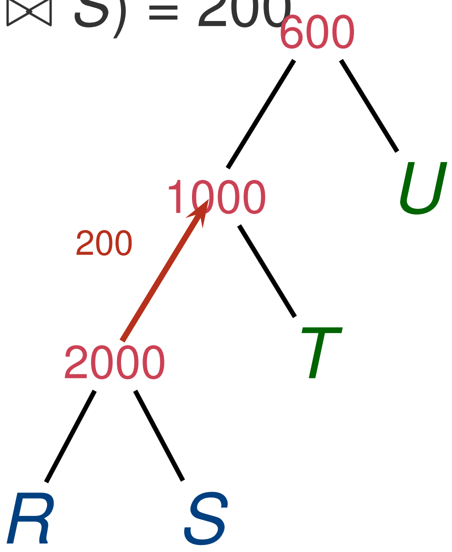
Plan 1: Total Cost = 4300

Plan 2: Total Cost = 3800

Example (cont.)

- **Cost model:**
  - **JC(X,Y)** = CPU & I/O cost of joining relations X & Y
  - **CC(X)** = Communication cost of sending relation X from one site to another site
  - JC(R, S) = 2000, JC(T, U) = 2000
  - JC($R \bowtie S$, T) = 1000, JC($R \bowtie S \bowtie T$, U) = 600
  - JC($R \bowtie S$, $T \bowtie U$) = 100, CC($R \bowtie S$) = 200





Plan 1:
Response Time = 2300

Plan 2:
Response Time = 3800

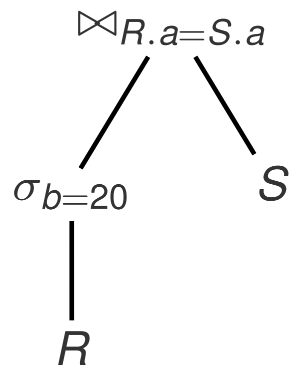
Query Processing Steps

- **Query rewriting**
  - Query decomposition
    - Translates query into relational algebra query
  - Data localization
    - Rewrites distributed query into a fragment query
- **Global query optimization**
  - Finds an optimal execution plan for query
- **Distributed query execution**
  - Executes query plan to compute query result

Query Decomposition

```
SELECT   *
FROM     R, S
WHERE    R.a = S.a
AND      R.b = 20
```

$\longrightarrow$



- **Normalization**
  - Rewrites query into some normal form
- **Semantic Analysis**
  - Checks that query is semantically correct
- **Simplification & Restructuring**
  - Rewrites query into simpler form (e.g., eliminates redundancy)

Normalization

- A simple predicate defined on a relation R is of the form "A_i op v" where A_i is an attribute of R, $op \in \{=, \neq, <, \leq, >, \geq\}$ and $v \in Domain(A_i)$
- **Conjunctive Normal Form (CNF)**

$$(p_{11} \vee p_{12} \cdots \vee p_{1n_1}) \wedge \cdots \wedge (p_{m1} \vee p_{m2} \cdots \vee p_{mn_m})$$

- **Disjunctive Normal Form (DNF)**

$$(p_{11} \wedge p_{12} \cdots \wedge p_{1n_1}) \vee \cdots \vee (p_{m1} \wedge p_{m2} \cdots \wedge p_{mn_m})$$

- Each p_{ij} is a simple predicate

Review of RA Equivalence Rules

attributes(R) = Set of attributes in schema of relation R
attributes(p) = Set of attributes in predicate p

1. **Commutativity of binary operators**
   1.1 $R \times S \equiv S \times R$
   1.2 $R \bowtie S \equiv S \bowtie R$
2. **Associativity of binary operators**
   2.1 $(R \times S) \times T \equiv R \times (S \times T)$
   2.2 $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$
3. **Idempotence of unary operators**
   3.1 $\pi_{L'}(\pi_L(R)) \equiv \pi_{L'}(R)$
   if $L' \subseteq L \subseteq attributes(R)$
   3.2 $\sigma_{p_1}(\sigma_{p_2}(R)) \equiv \sigma_{p_1 \wedge p_2}(R)$

Review of RA Equivalence Rules (cont.)

4. **Commutating selection with projection**
   4.1 $\pi_L(\sigma_p(R)) \equiv \pi_L(\sigma_p(\pi_{L \cup attributes(p)}(R)))$
5. **Commutating selection with binary operators**
   5.1 $\sigma_p(R \times S) \equiv \sigma_p(R) \times S$
   if $attributes(p) \subseteq attributes(R)$
   5.2 $\sigma_p(R \bowtie_{p'} S) \equiv \sigma_p(R) \bowtie_{p'} S$
   if $attributes(p) \subseteq attributes(R)$
   5.3 $\sigma_p(R \cup S) \equiv \sigma_p(R) \cup \sigma_p(S)$

Review of RA Equivalence Rules (cont.)

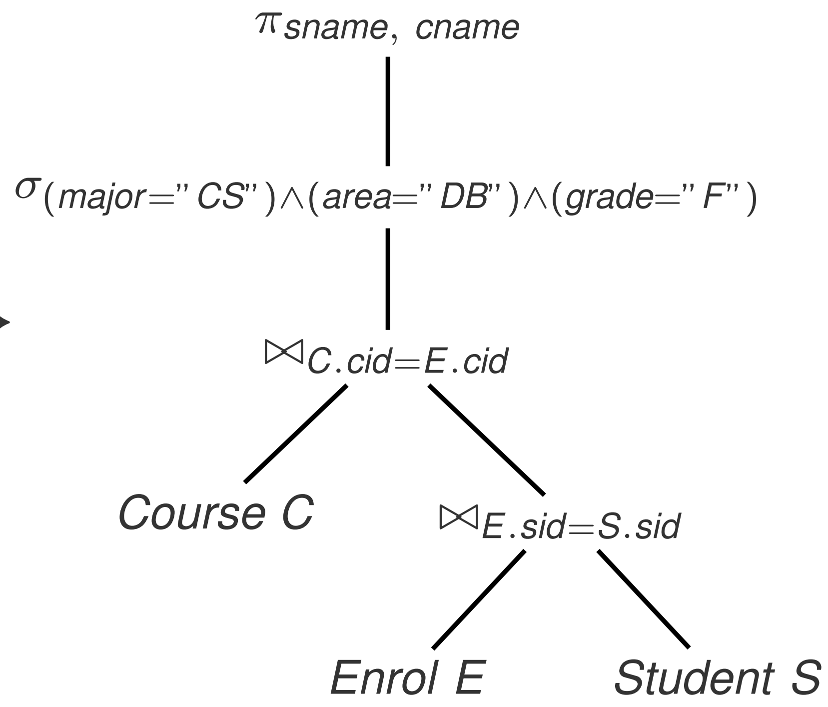
6. **Commutating projection with binary operators**
   Let $L = L_R \cup L_S$, where $L_R \subseteq attributes(R)$ and $L_S \subseteq attributes(S)$
   6.1 $\pi_L(R \times S) \equiv \pi_{L_R}(R) \times \pi_{L_S}(S)$
   6.2 $\pi_L(R \bowtie_p S) \equiv \pi_{L_R}(R) \bowtie_p \pi_{L_S}(S)$
   if $attributes(p) \cap attributes(R) \subseteq L_R$ and $attributes(p) \cap attributes(S) \subseteq L_S$
   6.3 $\pi_L(R \cup S) \equiv \pi_L(R) \cup \pi_L(S)$

Example

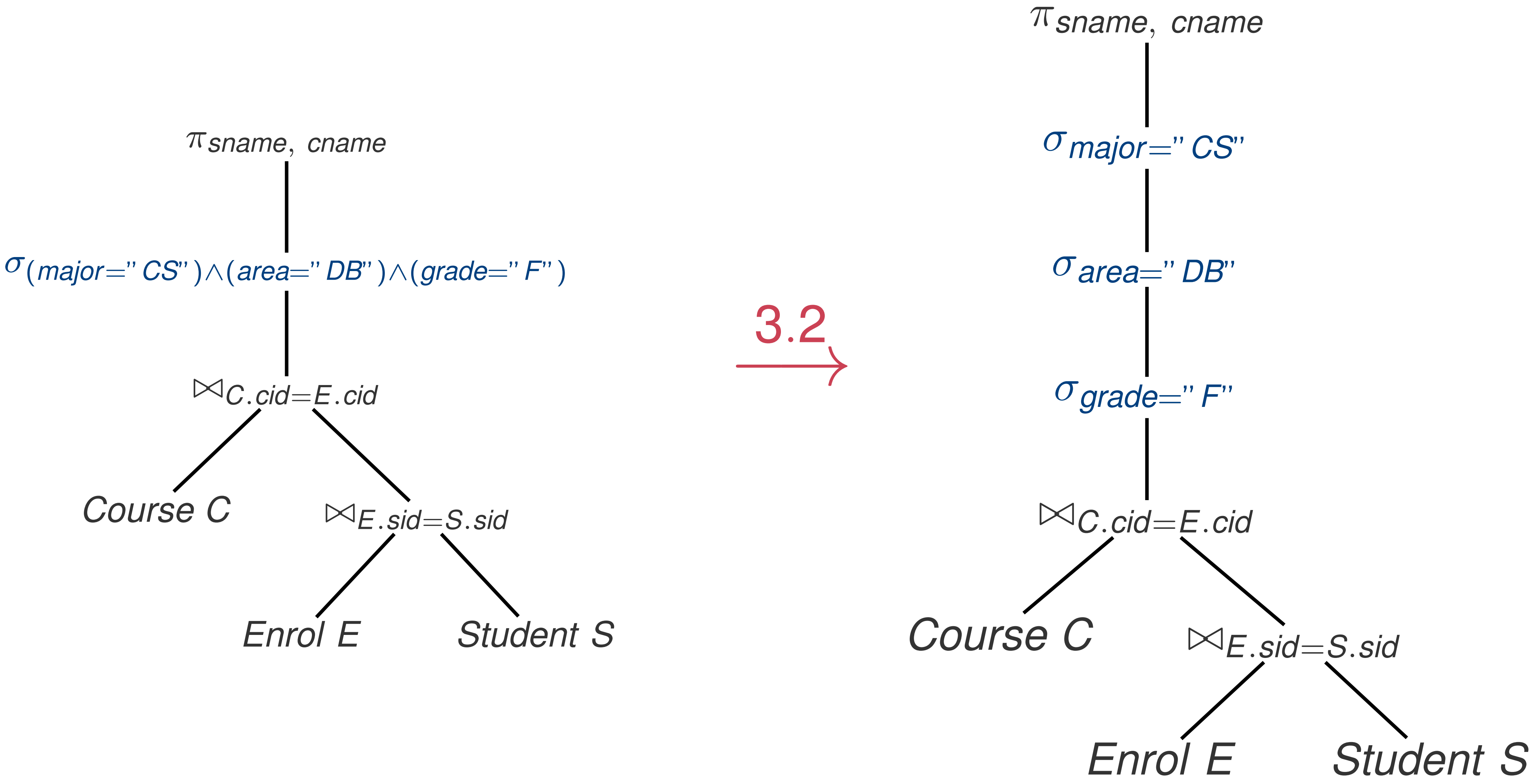
`Student` ($\underline{\text{sid}}$, sname, major)
`Course` ($\underline{\text{cid}}$, cname, area)
`Enrol` ($\underline{\text{sid,cid}}$, grade)

```
SELECT   sname, cname
FROM     Student S, Course C, Enrol E
WHERE    E.sid = S.sid
AND      E.cid = C.cid
AND      major = "CS"
AND      area = "DB"
AND      grade = "F"
```

$\longrightarrow$

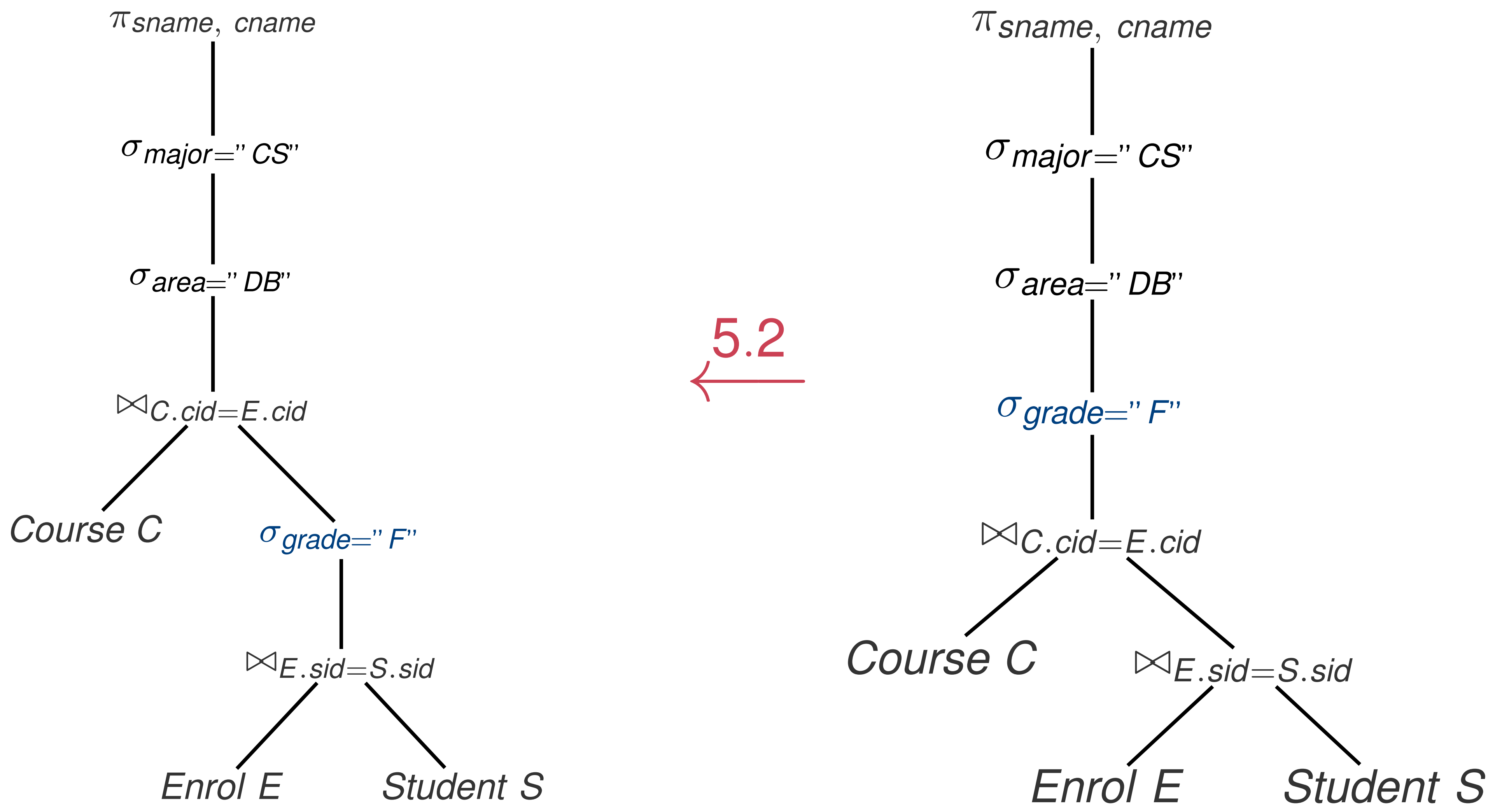


Example (cont.)



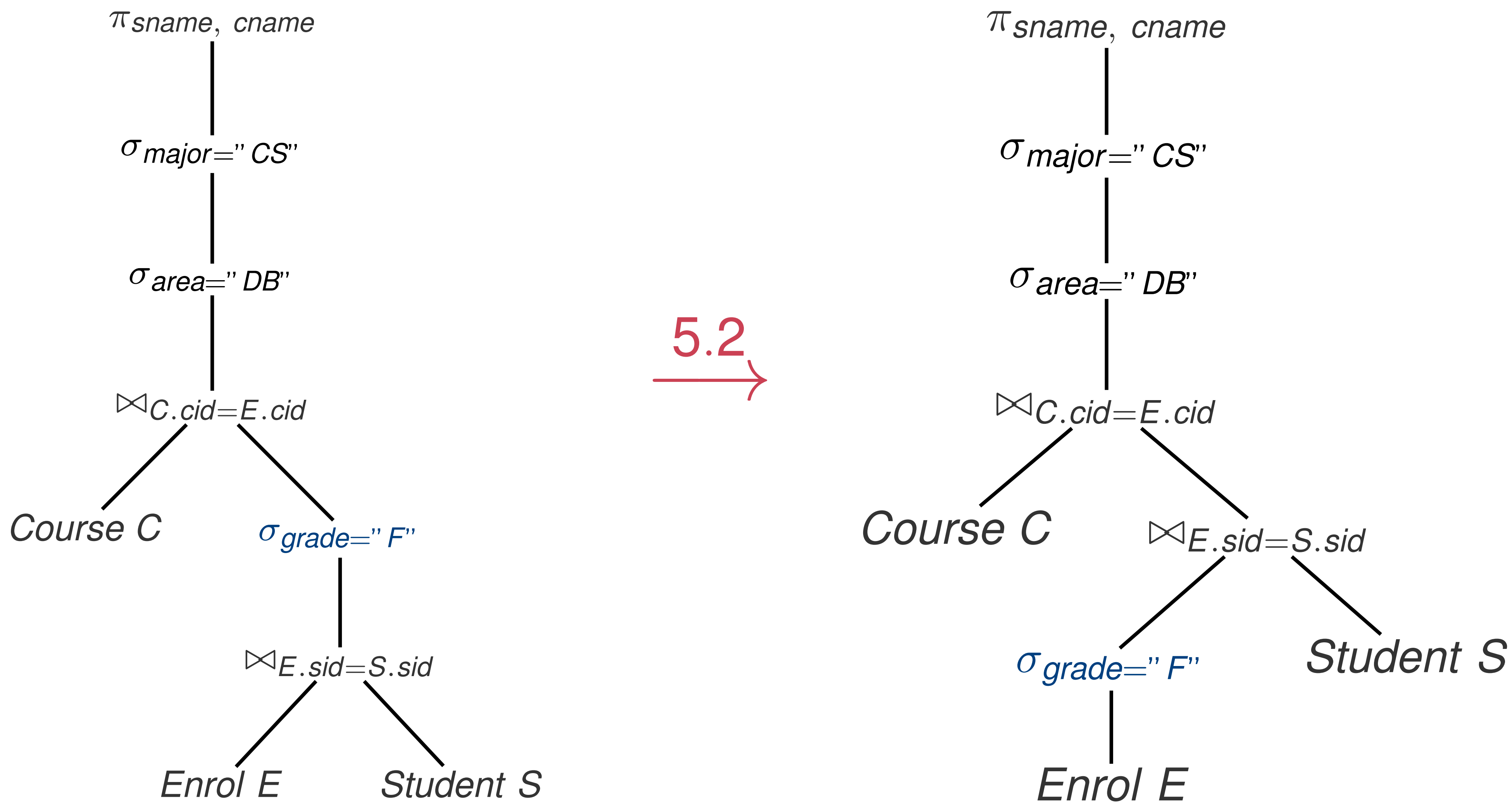
$$\sigma_{p_1}(\sigma_{p_2}(R)) \equiv \sigma_{p_1 \wedge p_2}(R)$$

Example (cont.)



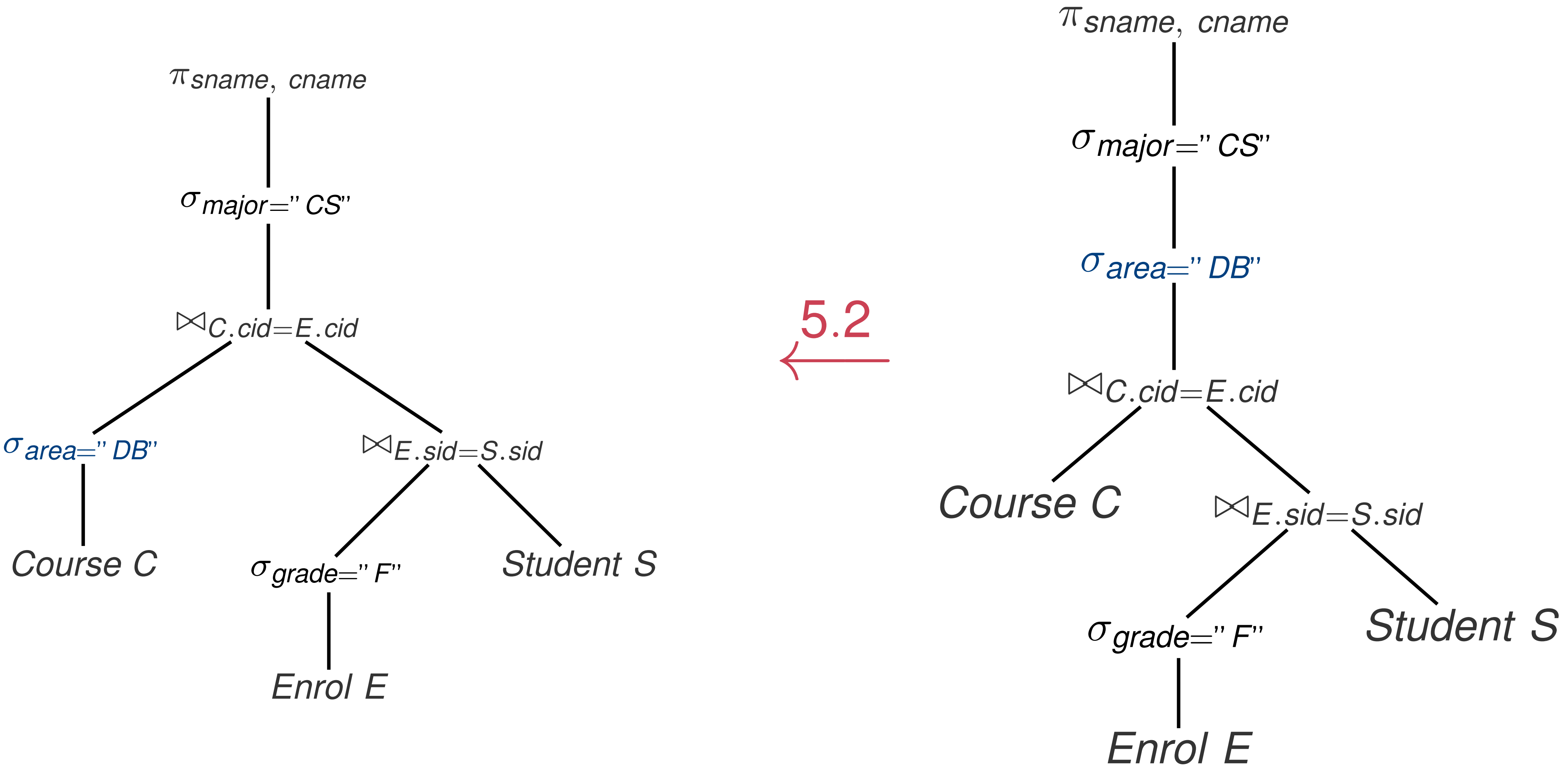
$$\sigma_p(R \bowtie_{p'} S) \equiv \sigma_p(R) \bowtie_{p'} S \text{ if } attributes(p) \subseteq attributes(R)$$

Example (cont.)



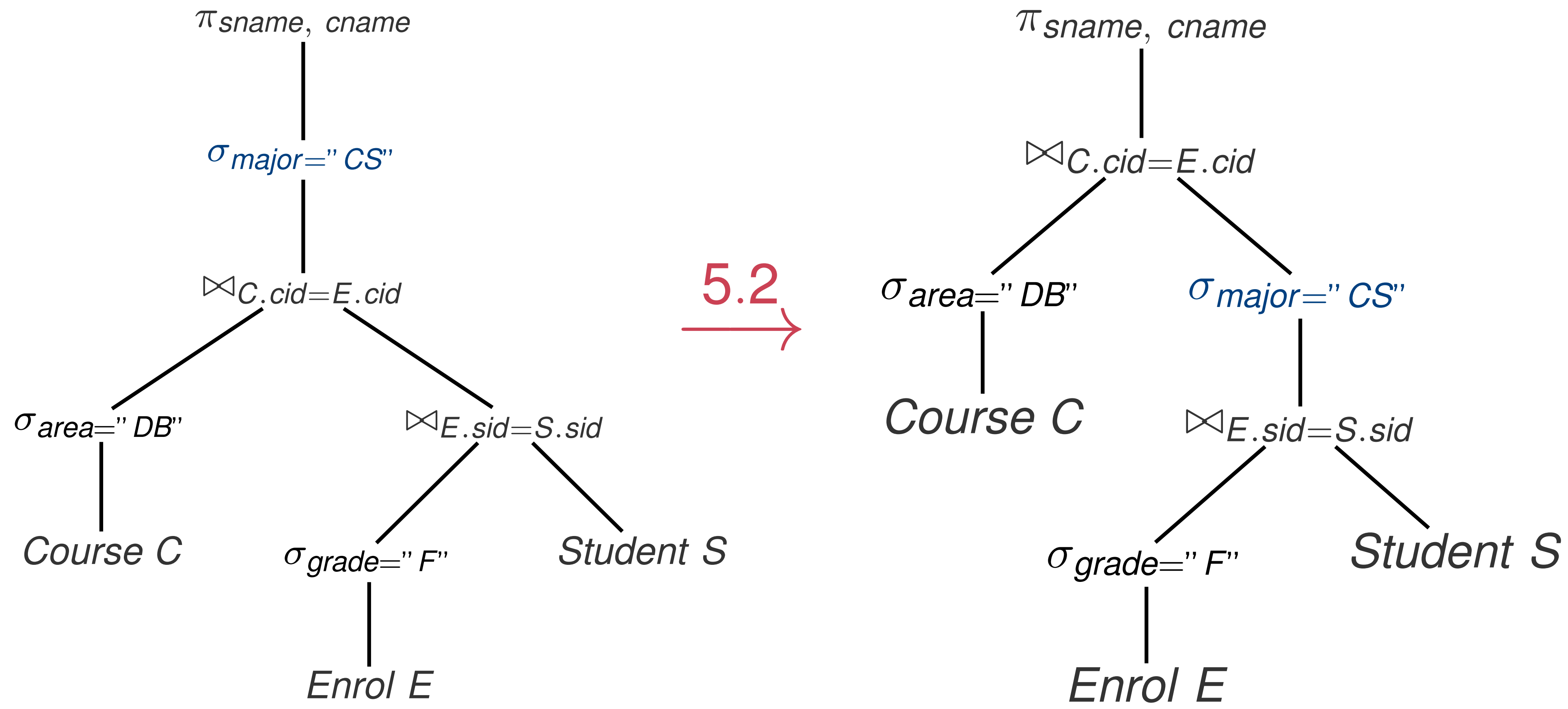
$\sigma_p(R \bowtie_{p'} S) \equiv \sigma_p(R) \bowtie_{p'} S$ if $attributes(p) \subseteq attributes(R)$

Example (cont.)



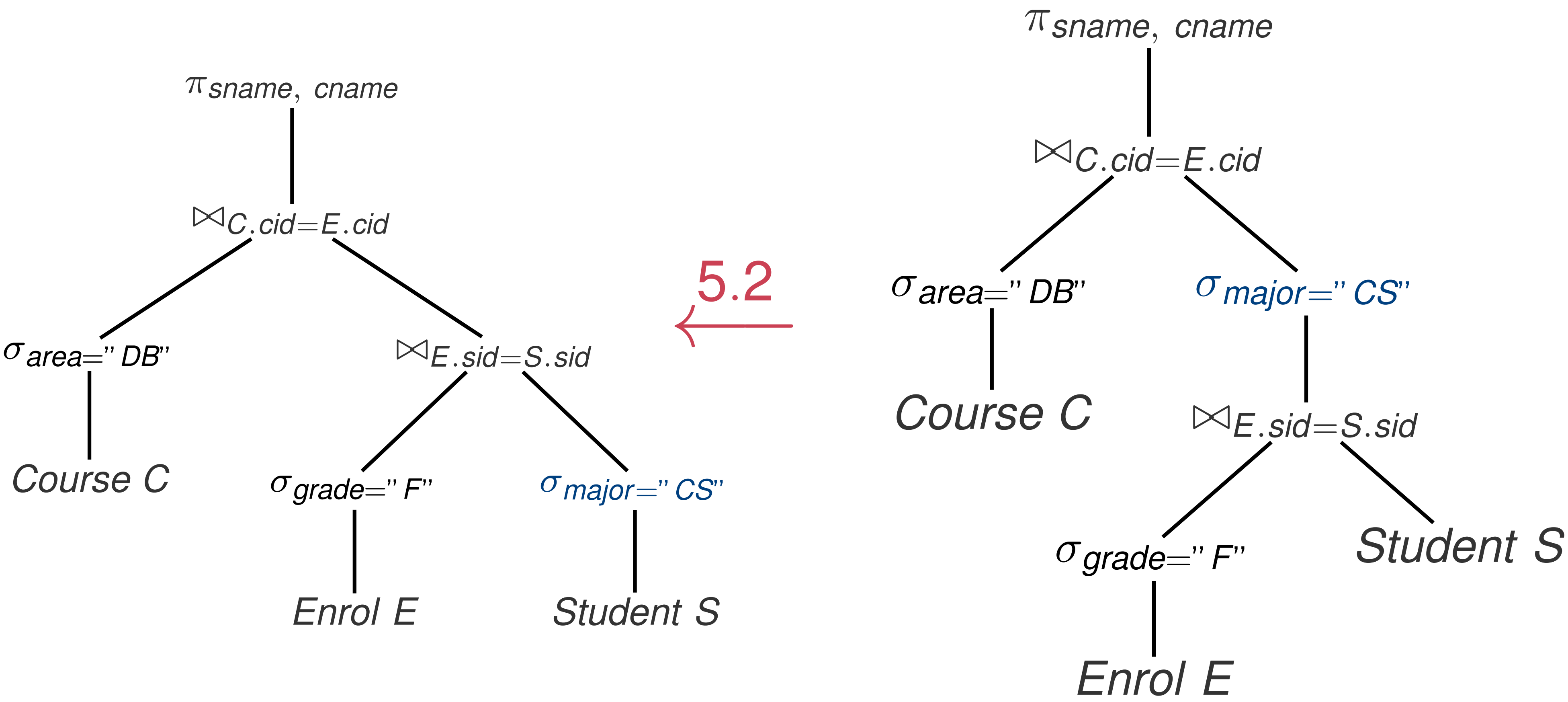
$\sigma_p(R \bowtie_{p'} S) \equiv \sigma_p(R) \bowtie_{p'} S$ if $attributes(p) \subseteq attributes(R)$

Example (cont.)



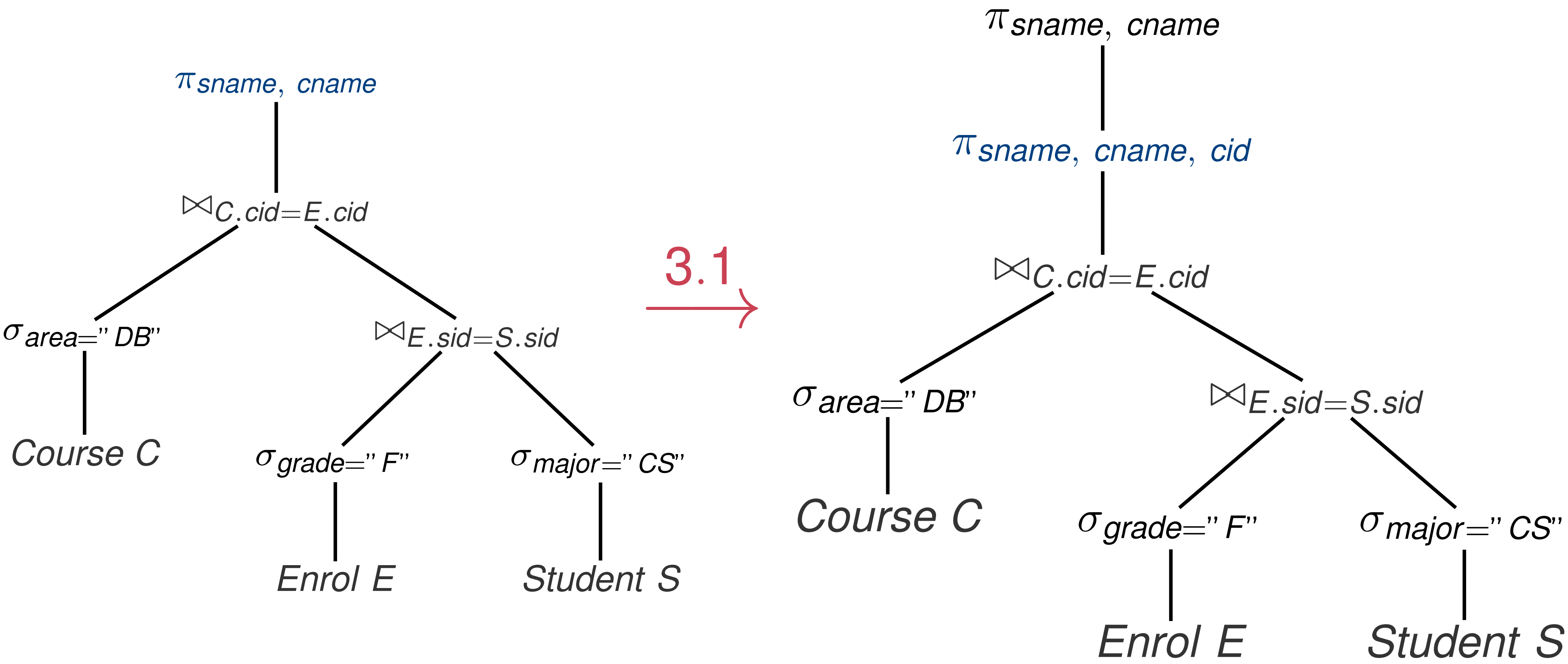
$$\sigma_p(R \bowtie_{p'} S) \equiv \sigma_p(R) \bowtie_{p'} S \text{ if } attributes(p) \subseteq attributes(R)$$

Example (cont.)



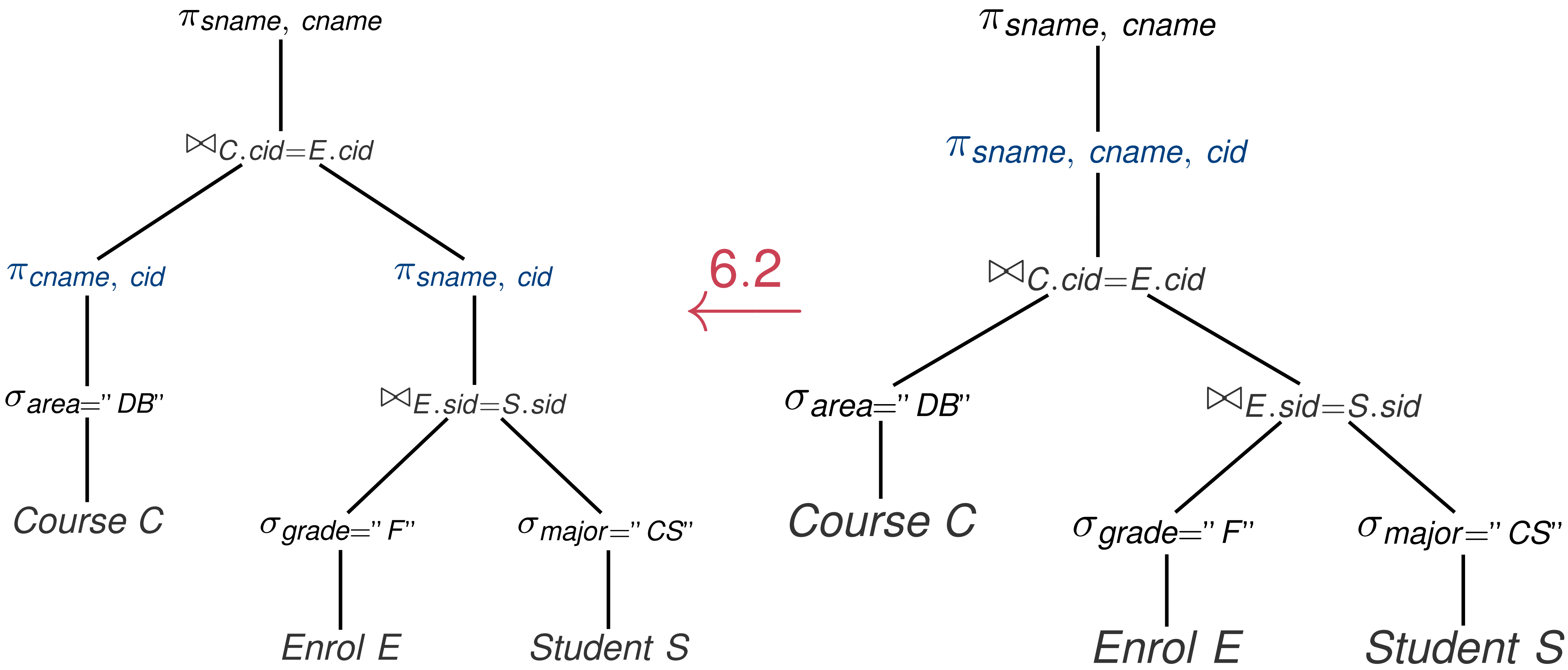
$$\sigma_p(R \bowtie_{p'} S) \equiv \sigma_p(R) \bowtie_{p'} S \text{ if } attributes(p) \subseteq attributes(R)$$

Example (cont.)



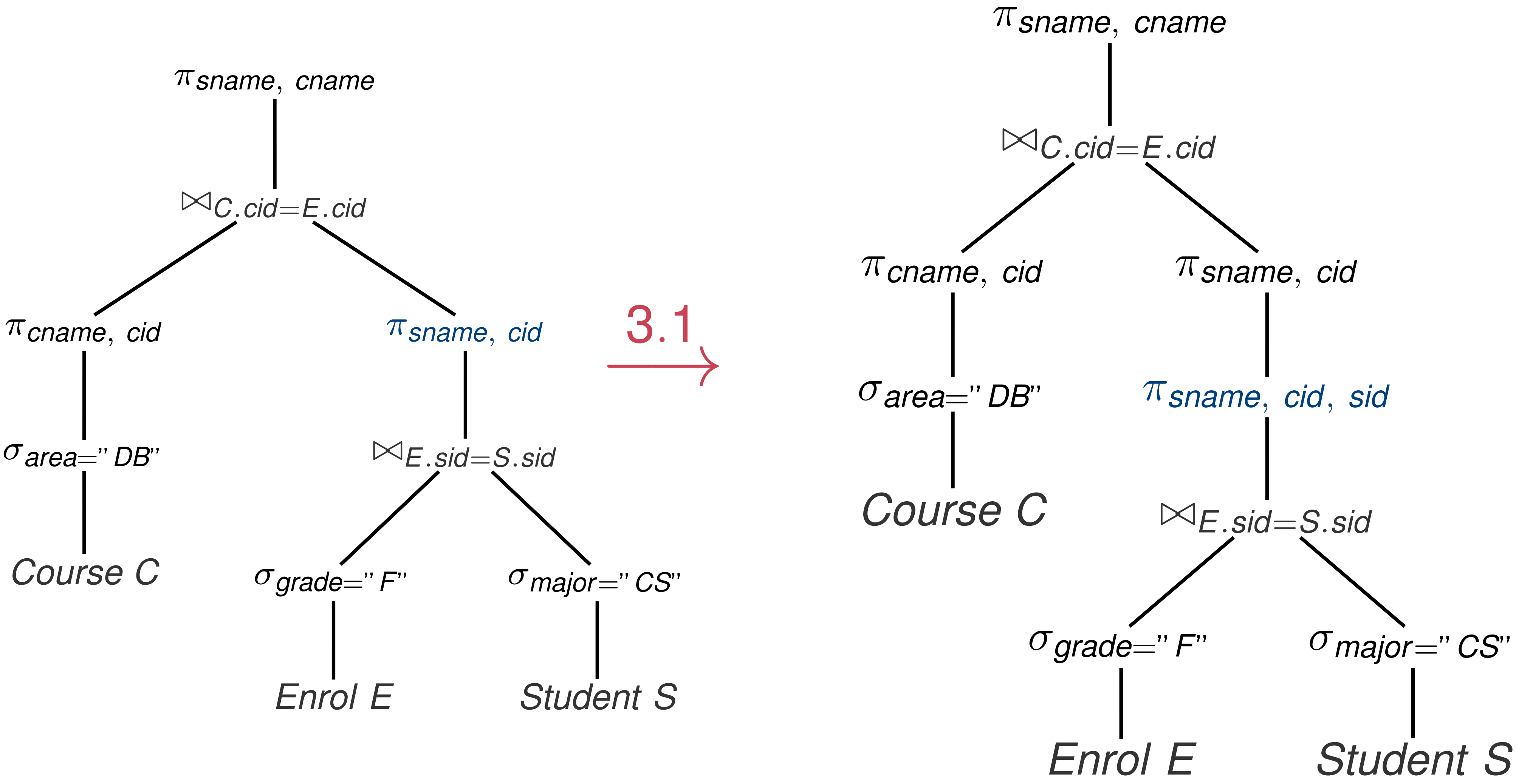
$$\pi_{L'}(\pi_L(R)) \equiv \pi_{L'}(R) \text{ if } L' \subseteq L \subseteq attributes(R)$$

Example (cont.)



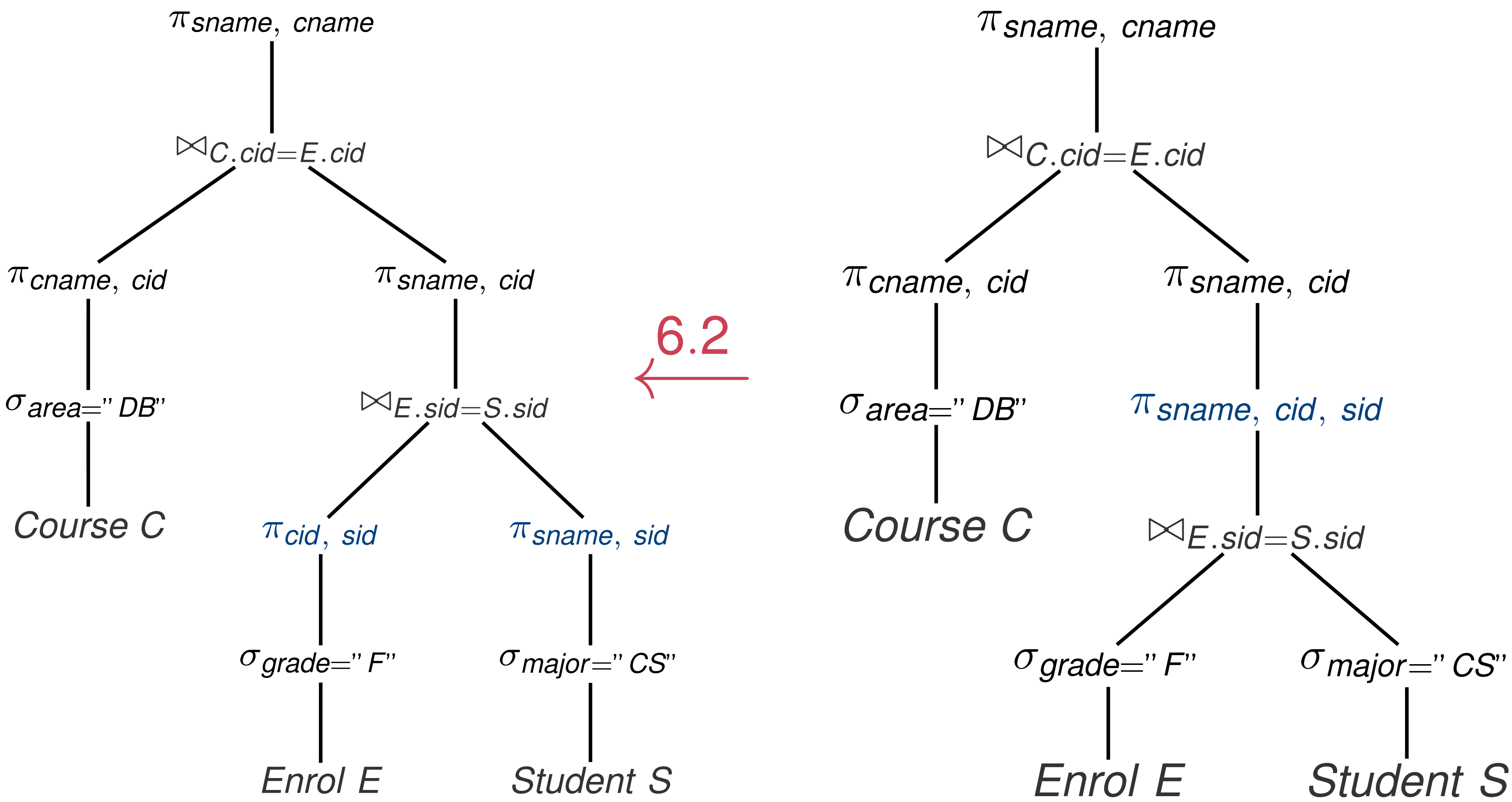
$$\pi_L(R \bowtie_p S) \equiv \pi_{L_R}(R) \bowtie_p \pi_{L_S}(S)$$

Example (cont.)



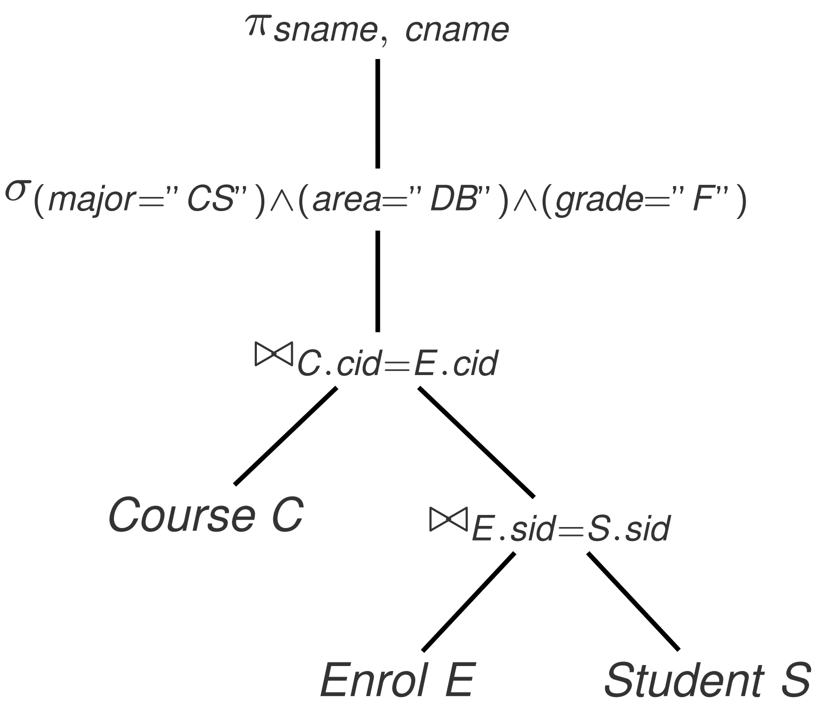
$$\pi_{L'}(\pi_L(R)) \equiv \pi_{L'}(R) \text{ if } L' \subseteq L \subseteq attributes(R)$$

Example (cont.)

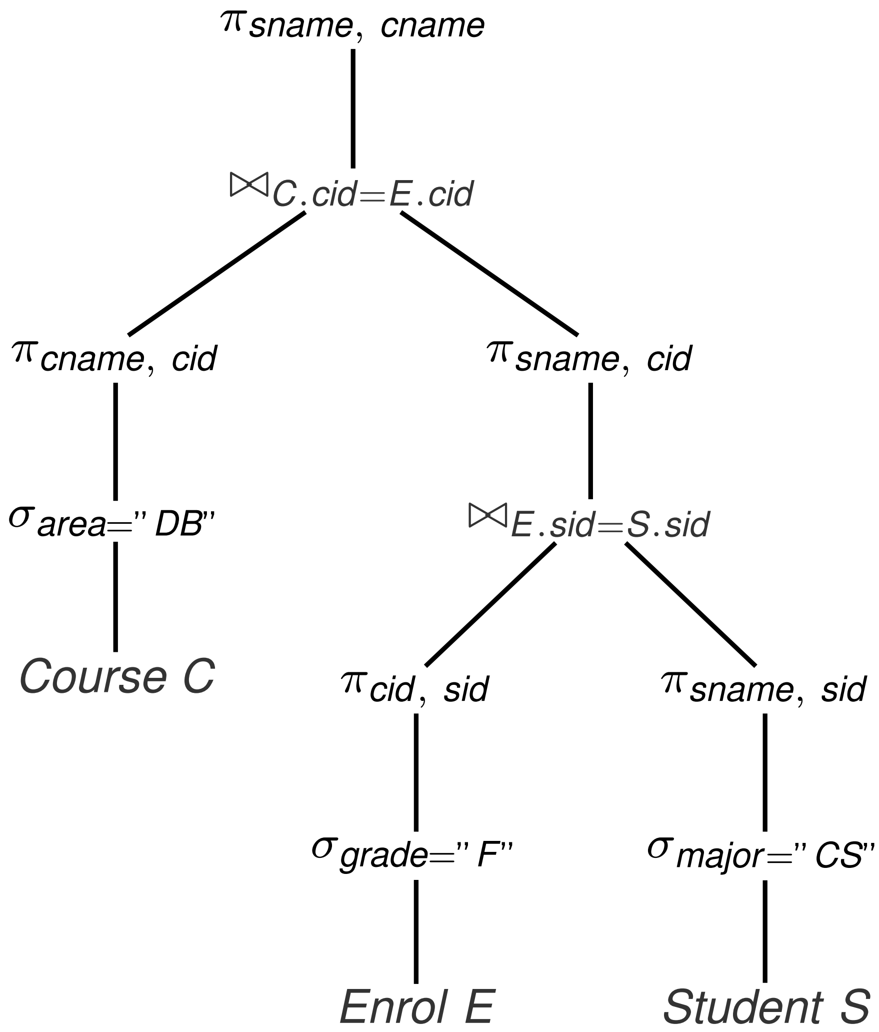


$$\pi_L(R \bowtie_p S) \equiv \pi_{L_R}(R) \bowtie_p \pi_{L_S}(S)$$

Example (cont.)



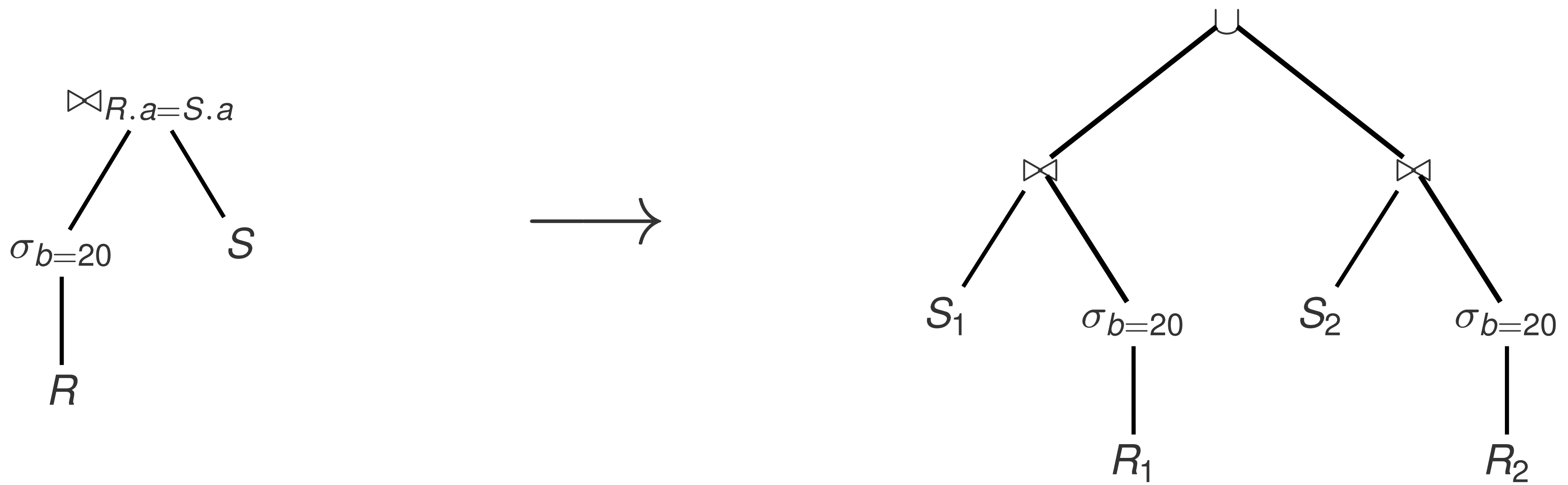
Original Query



Final Query

Query Localization

- Rewrites distributed query into a fragment query
- Uses data distribution information to determine which fragments are involved



Distributed Query

Fragment Query

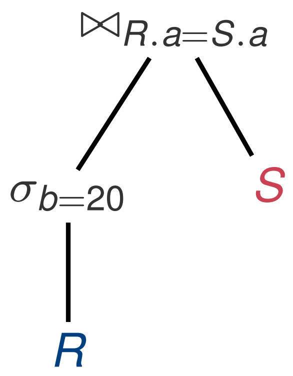
Localization Program

- A localization program for a fragmented relation R is a reconstruction rule for R in terms of its fragments
- **Example**:
  - Let $\{R_1, \cdots, R_n\}$ be a complete & disjoint partitioning of R
  - If each $R_i = \sigma_{F_i}(R)$, then localization program for R is $R_1 \cup \cdots \cup R_n$
  - If each $R_i = \pi_{L_i}(R)$ & $key(R) \in L_i$, then localization program for R is $R_1 \bowtie \cdots \bowtie R_n$

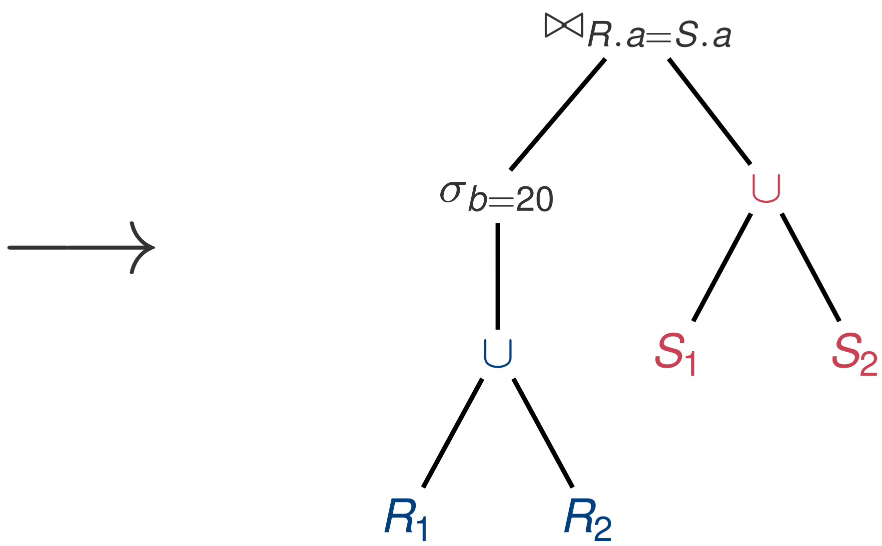
Localized Query

- Localized query = query with each fragmented relation replaced by its localization program

$R_1 = \sigma_{a \leq 10}(R), \;\; R_2 = \sigma_{a > 10}(R)$
$S_1 = \sigma_{a \leq 10}(S), \;\; S_2 = \sigma_{a > 10}(S)$



Distributed Query



Localized Query

Reduction Techniques

- Reduction techniques = rewriting techniques to simplify localized queries
- Identify & eliminate query expressions on fragments that do not contribute to query results
- Techniques:
  - Reduction for horizontal fragmentation
    - Reduction with selection
    - Reduction with join
  - Reduction for derived horizontal fragmentation
  - Reduction for vertical fragmentation
  - Reduction for hybrid fragmentation

Reduction with Selection

Rule 1: $\sigma_p(R_i) = \emptyset$ if $R_i = \sigma_{F_i}(R)$ and $F_i \wedge p = \mathit{false}$

$$
\begin{aligned}
R &= R_1 \cup R_2 \cup R_3 \\
& \quad R_1 = \sigma_{a<10}(R) \\
& \quad R_2 = \sigma_{a\in[10,70]}(R) \\
& \quad R_3 = \sigma_{a>70}(R) \\
\\
Q_1 &= \sigma_{a=12}(R) \\
&= \sigma_{a=12}(R_1 \cup R_2 \cup R_3) \\
&= \sigma_{a=12}(R_1) \cup \sigma_{a=12}(R_2) \cup \sigma_{a=12}(R_3) \\
&= \sigma_{a=12}(R_2)
\end{aligned}
$$

Reduction with Join

Rule 2: $R_i \bowtie_a S_j = \emptyset$ if $R_i = \sigma_{F_a \wedge F}(R)$, $S_j = \sigma_{F'_a \wedge F'}(S)$, $F_a \& F'_a$ are predicates on attribute a, and $F_a \wedge F'_a = false$

$$
\begin{aligned}
R &= R_1 \cup R_2 \cup R_3 \\
& R_1 = \sigma_{a<10}(R) \\
& R_2 = \sigma_{a \in [10,70]}(R) \\
& R_3 = \sigma_{a>70}(R)
\end{aligned}
\qquad
\begin{aligned}
S &= S_1 \cup S_2 \\
& S_1 = \sigma_{a<10}(S) \\
& S_2 = \sigma_{a \geq 10}(S)
\end{aligned}
$$

$$
\begin{aligned}
Q &= R \bowtie_a S \\
&= (R_1 \cup R_2 \cup R_3) \bowtie_a (S_1 \cup S_2) \\
&= (R_1 \bowtie_a S_1) \cup (R_1 \bowtie_a S_2) \cup (R_2 \bowtie_a S_1) \cup \\
&\quad (R_2 \bowtie_a S_2) \cup (R_3 \bowtie_a S_1) \cup (R_3 \bowtie_a S_2) \\
&= (R_1 \bowtie_a S_1) \cup (R_2 \bowtie_a S_2) \cup (R_3 \bowtie_a S_2)
\end{aligned}
$$

Reduction for Derived Fragmentation

Rule 3: $S_i \bowtie_a R_j = \emptyset$ if $S_i = S \ltimes_a R_i$ is a derived horizontal fragmentation of S wrt R, and $i \neq j$

Consider $R(\underline{a}, b, c)$, $S(\underline{x}, y, a)$
where $S.a$ is a foreign key of R

- $R_1 = \sigma_{b=10}(R)$, $\quad R_2 = \sigma_{b \neq 10}(R)$
- $S_1 = S \ltimes_a R_1$, $\quad S_2 = S \ltimes_a R_2$

$$\begin{aligned}
Q &= \sigma_{b=20}(R) \bowtie_a S \\
&= \sigma_{b=20}(R_1 \cup R_2) \bowtie_a (S_1 \cup S_2) \\
&= \sigma_{b=20}(R_2) \bowtie_a (S_1 \cup S_2) \\
&= (\sigma_{b=20}(R_2) \bowtie_a S_1) \cup (\sigma_{b=20}(R_2) \bowtie_a S_2) \\
&= \sigma_{b=20}(R_2) \bowtie_a S_2
\end{aligned}$$

Reduction for Vertical Fragmentation

Rule 4: $\pi_L(R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n) = \pi_L(R_2 \bowtie \cdots \bowtie R_n)$ if $R_1, \cdots, R_n$ are vertical fragments of R and $(attributes(R_1) - key(R)) \cap L = \emptyset$

Consider R($\underline{a}$,b,c) where

$$\begin{aligned} R &= R_1 \bowtie_a R_2 \\ & \quad R_1 = \pi_{a,b}(R) \\ & \quad R_2 = \pi_{a,c}(R) \end{aligned}$$

$$\begin{aligned} Q &= \pi_c(R) \\ &= \pi_c(R_1 \bowtie_a R_2) \\ &= \pi_c(R_2) \end{aligned}$$

Reduction for Hybrid Fragmentation

$$
\begin{aligned}
R &= (R_1 \cup R_2) \bowtie_a R_3 \\
& \quad R_1 = \pi_{a,b}(\sigma_{a<10}(R)) \\
& \quad R_2 = \pi_{a,b}(\sigma_{a\geq 10}(R)) \\
& \quad R_3 = \pi_{a,c}(R) \\
\\
Q &= \pi_b(\sigma_{a=20}(R)) \\
&= \pi_b(\sigma_{a=20}((R_1 \cup R_2) \bowtie_a R_3)) \\
&= \pi_b(\sigma_{a=20}(R_1 \cup R_2)) \\
&= \pi_b(\sigma_{a=20}(R_1) \cup \sigma_{a=20}(R_2)) \\
&= \pi_b(\sigma_{a=20}(R_2))
\end{aligned}
$$

Distributed Join Strategies for $R \bowtie_A S$

- There are three cases to consider for $R \bowtie_A S$:
  - Case 1: Both R and S have been partitioned on join key
  - Case 2: Only R (but not S) has been partitioned on join key
  - Case 3: Neither R nor S has been partitioned on join key
- Case 1: Collocated/Local join
- Case 2: Directed join
  - Dynamically repartition S on join key
- Case 3: Repartitioned join
  - Dynamically repartition R & S on join key
- Broadcast join: Replicate either R or S to all nodes
  - Applicable for cases 2 & 3

Join Strategies: Example

- Customers (<u>cust#</u>, cname, city)
- Orders (<u>order#</u>, cust#, odate)
- Suppliers (<u>supp#</u>, sname, city)

Customers

cust#	cname	city
1	Alice	Singapore
2	Bob	Penang
3	Carol	Bangkok
4	Dave	Singapore
5	Eve	Singapore
6	Fred	Penang
7	George	Bangkok

Orders

order#	cust#	odate
302	1	June 2013
304	2	May 2013
307	3	Nov 2013
308	1	April 2013
309	5	May 2013
311	6	Dec 2013
312	3	July 2013

Suppliers

supp#	sname	city
32	A	Bangkok
33	B	Singapore
34	C	Singapore
36	D	Bangkok
37	E	Penang
38	F	Penang
39	G	Singapore

Collocated Join: Example

- Customers hash partitioned on cust# using h_{cust}
- Orders hash partitioned on cust# using h_{cust}
- $h_{cust}(c) = (c \mod 3) + 1$

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Orders$_1$

order#	cust#	odate
307	3	Nov 2013
311	6	Dec 2013
312	3	July 2013

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Orders$_2$

order#	cust#	odate
302	1	June 2013
308	1	April 2013

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Orders$_3$

order#	cust#	odate
304	2	May 2013
309	5	May 2013

Collocated Join: Example (cont.)

- **Query 1**: Customers $\bowtie_{cust\#}$ Orders

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Orders$_1$

order#	cust#	odate
307	3	Nov 2013
311	6	Dec 2013
312	3	July 2013

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Orders$_2$

order#	cust#	odate
302	1	June 2013
308	1	April 2013

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Orders$_3$

order#	cust#	odate
304	2	May 2013
309	5	May 2013

$$\bigcup_{i=1}^{3}(Customers_i \bowtie Orders_i)$$

Directed Join: Example

- Customers hash partitioned on cust# using h_{cust}
- Orders hash partitioned on order# using h_{order}
- $h_{cust}(c) = (c \mod 3) + 1$
- $h_{order}(o) = (o \mod 3) + 1$

Site 1

$Customers_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

$Orders_1$

order#	cust#	odate
309	5	May 2013
312	3	July 2013

Site 2

$Customers_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

$Orders_2$

order#	cust#	odate
304	2	May 2013
307	3	Nov 2013

Site 3

$Customers_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

$Orders_3$

order#	cust#	odate
302	1	June 2013
308	1	April 2013
311	6	Dec 2013

Directed Join: Example (cont.)

- **Query 1**: Customers $\bowtie_{cust\#}$ Orders

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Orders$_1$

order#	cust#	odate
309	5	May 2013
312	3	July 2013

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Orders$_2$

order#	cust#	odate
304	2	May 2013
307	3	Nov 2013

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Orders$_3$

order#	cust#	odate
302	1	June 2013
308	1	April 2013
311	6	Dec 2013

Directed Join: Example (cont.)

- **Query 1**: Customers $\bowtie_{cust\#}$ Orders

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Orders$_1$

order#	cust#	odate
309	5	May 2013
312	3	July 2013

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Orders$_2$

order#	cust#	odate
304	2	May 2013
307	3	Nov 2013

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Orders$_3$

order#	cust#	odate
302	1	June 2013
308	1	April 2013
311	6	Dec 2013

Repartition Orders on cust# : $Orders'_i = \sigma_{h_{cust}(cust\#)=i}(Orders)$

$$\bigcup_{i=1}^{3}(Customers_i \bowtie Orders'_i)$$

Repartitioned Join: Example

- Customers hash partitioned on cust# using h_{cust}
- Suppliers hash partitioned on supp# using h_{supp}
- $h_{cust}(c) = (c \mod 3) + 1$
- $h_{supp}(s) = (s \mod 3) + 1$

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Suppliers$_1$

supp#	sname	city
33	B	Singapore
36	D	Bangkok
39	G	Singapore

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Suppliers$_2$

supp#	sname	city
34	C	Singapore
37	E	Penang

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Suppliers$_3$

supp#	sname	city
32	A	Bangkok
38	F	Penang

Repartitioned Join: Example (cont.)

- **Query 2:** Customers $\bowtie_{city}$ Suppliers

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Suppliers$_1$

supp#	sname	city
33	B	Singapore
36	D	Bangkok
39	G	Singapore

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Suppliers$_2$

supp#	sname	city
34	C	Singapore
37	E	Penang

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Suppliers$_3$

supp#	sname	city
32	A	Bangkok
38	F	Penang

Repartitioned Join: Example (cont.)

- **Query 2**: Customers $\bowtie_{city}$ Suppliers
- Repartition both tables using h_{city}

c	$h_{city}(c)$
Singapore	1
Penang	2
Bangkok	3

- $Customers'_i = \sigma_{h_{city}(city)=i}(Customers)$
- $Suppliers'_i = \sigma_{h_{city}(city)=i}(Suppliers)$

Repartitioned Join: Example (cont.)

Query 2: Customers $\bowtie_{city}$ Suppliers

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Suppliers$_1$

supp#	sname	city
33	B	Singapore
36	D	Bangkok
39	G	Singapore

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Suppliers$_2$

supp#	sname	city
34	C	Singapore
37	E	Penang

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Suppliers$_3$

supp#	sname	city
32	A	Bangkok
38	F	Penang

$$\bigcup_{i=1}^{3} (Customers_i' \bowtie Suppliers_i')$$

Broadcast Join: Example

- Customers hash partitioned on cust# using h_{cust}
- Suppliers hash partitioned on supp# using h_{supp}
- $h_{cust}(c) = (c \mod 3) + 1$
- $h_{supp}(s) = (s \mod 3) + 1$

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Suppliers$_1$

supp#	sname	city
33	B	Singapore
36	D	Bangkok
39	G	Singapore

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Suppliers$_2$

supp#	sname	city
34	C	Singapore
37	E	Penang

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Suppliers$_3$

supp#	sname	city
32	A	Bangkok
38	F	Penang

Broadcast Join: Example (cont.)

- **Query 2:** Customers $\bowtie_{city}$ Suppliers

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Suppliers$_1$

supp#	sname	city
33	B	Singapore
36	D	Bangkok
39	G	Singapore

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Suppliers$_2$

supp#	sname	city
34	C	Singapore
37	E	Penang

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Suppliers$_3$

supp#	sname	city
32	A	Bangkok
38	F	Penang

Broadcast Join: Example (cont.)

Query 2: Customers $\bowtie_{city}$ Suppliers

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Suppliers$_1$

supp#	sname	city
33	B	Singapore
36	D	Bangkok
39	G	Singapore

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Suppliers$_2$

supp#	sname	city
34	C	Singapore
37	E	Penang

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Suppliers$_3$

supp#	sname	city
32	A	Bangkok
38	F	Penang

Option 1: Broadcast Suppliers

$$\bigcup_{i=1}^{3} (Customers_i \bowtie Suppliers)$$

Broadcast Join: Example (cont.)

- **Query 2**: Customers $\bowtie_{city}$ Suppliers

Site 1

Customers$_1$

cust#	cname	city
3	Carol	Bangkok
6	Fred	Penang

Suppliers$_1$

supp#	sname	city
33	B	Singapore
36	D	Bangkok
39	G	Singapore

Site 2

Customers$_2$

cust#	cname	city
1	Alice	Singapore
4	Dave	Singapore
7	George	Bangkok

Suppliers$_2$

supp#	sname	city
34	C	Singapore
37	E	Penang

Site 3

Customers$_3$

cust#	cname	city
2	Bob	Penang
5	Eve	Singapore

Suppliers$_3$

supp#	sname	city
32	A	Bangkok
38	F	Penang

Option 2: Broadcast Customers

$$\bigcup_{i=1}^{3} (Customers \bowtie Suppliers_i)$$

Comparison of join strategies for $R \bowtie S$

Let the relations be partitioned over n nodes

Join Strategy	**Communication Cost**
Collocated	0
Directed	$size(R)$ if R is being repartitioned
Repartitioned	$size(R) + size(S)$
Broadcast	$(n-1) \times size(R)$ if R is being broadcast

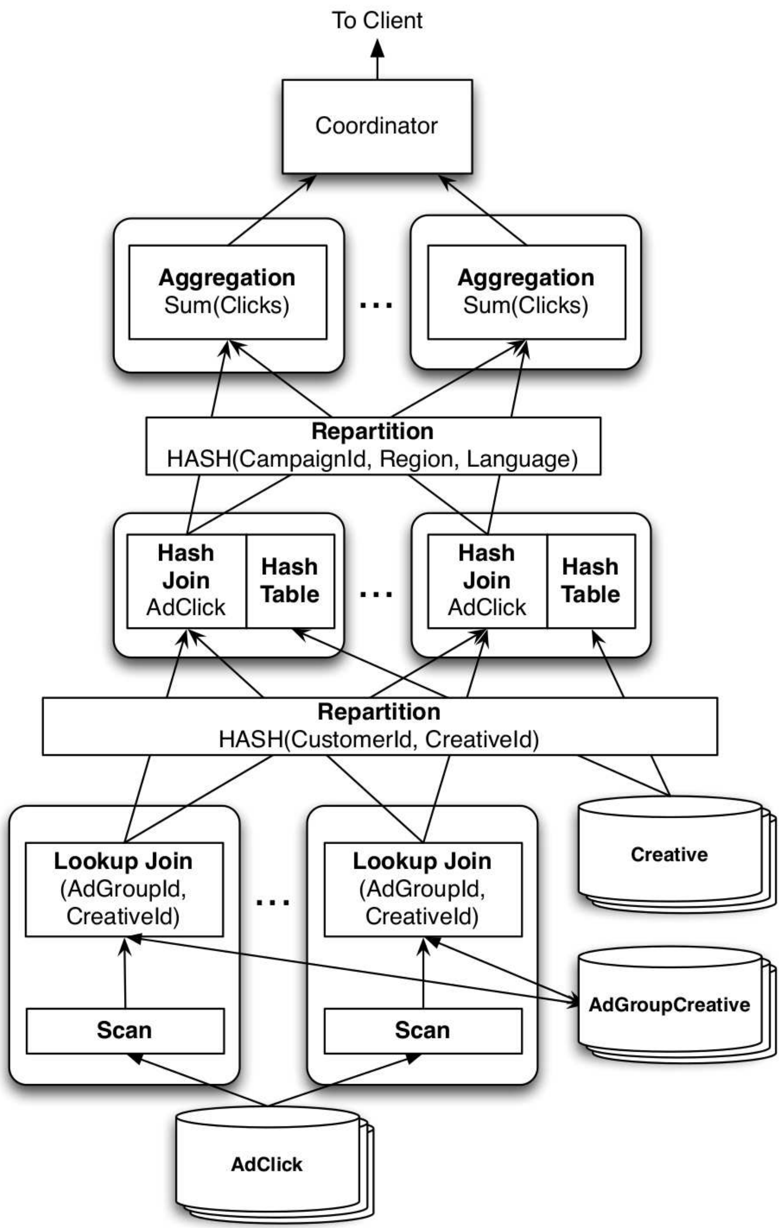
Query Processing in Google's F1

- Google's early NewSQL system
  - Distributed relational database system
  - Hybrid of NoSQL & RDBMS
  - NoSQL: High availability, scalability
  - RDBMS: Functionality, consistency, usability (SQL, transactions)
- Used by Google's AdWords system since 2012
  - 100s of applications & 1000s of users
  - Database is over 100 TB, 10^5 requests/sec

Query Processing in Google's F1

SELECT agc.CampaignId,
ac.Region,
c.Language,
SUM(ac.Clicks)
FROM AdClick ac
JOIN AdGroupCreative agc
USING (AdGroupId, CreativeId)
JOIN Creative c
USING (CustomerId, CreativeId)
WHERE ac.Date = '2013-02-23'
GROUP BY agc.CampaignId, ac.Region,
c.Language

Creative (CreativeId, CustomerId, Language, ...)
AdGroupCreative (AdGroupId, CreativeId, CampaignId, CustomerId, ...)
AdClick (AdGroupId, CreativeId, Region, Date, Clicks, ...)



(Shute, et al., 2013)

References

- T. Özsu & P. Valdureiz, *Distributed Query Processing*, Chapter 4, Principles of Distributed Database Systems, 4^{th} Edition, 2020
- C. Baru, et al. *DB2 Parallel Edition*, IBM Systems Journal, 34(2), 1995.
- *Google goes back to the future with SQL F1 database*, The Register, August 2013, https://www.theregister.co.uk/2013/08/30/google_f1_deepdive/
- *F1: A Distributed SQL Database That Scales*, VLDB 2013, https://research.google.com/pubs/pub41344.html