

Query Processing

CS5225

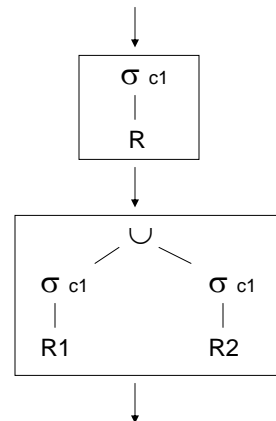
Query Processing

1

Query Processing

- Decomposition
- Localization
- Optimization
 - Global
 - Local

Select * From R Where c1



Optimal processing strategy

- Ship R1, R2 to result site, then select
- Perform select at R1 site, R2 site, then ship result

CS5225

Query Processing

2

Decomposition

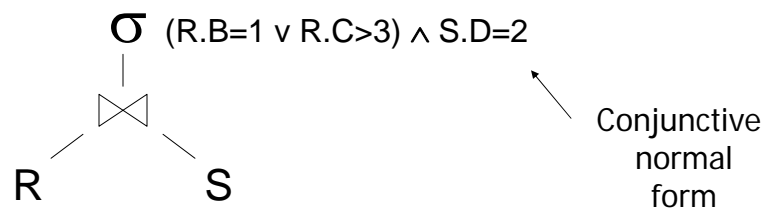
- Same as in centralized system
 - Normalization
 - Eliminating redundancy
 - Algebraic rewriting

Normalization

- Convert from general language to a “standard” form (e.g., Relational Algebra)

Example

Select A,C
From R,S
Where (R.B=1 and S.D=2) or (R.C>3 and S.D=2)



CS5225

Query Processing

5

Also: Detect invalid expressions

E.g.: Select * from R where R.A =3
☒ r does not have "A" attribute

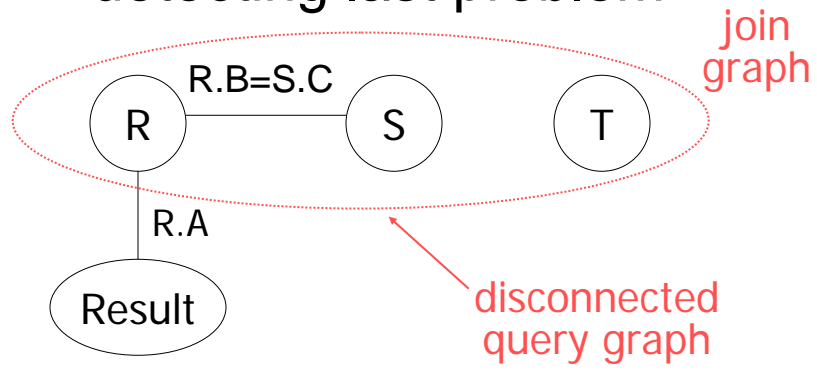
E.g.: Select R.A
From R, S, T
Where R.B=S.C

CS5225

Query Processing

6

Note: Query graph useful for detecting last problem



Eliminate redundancy

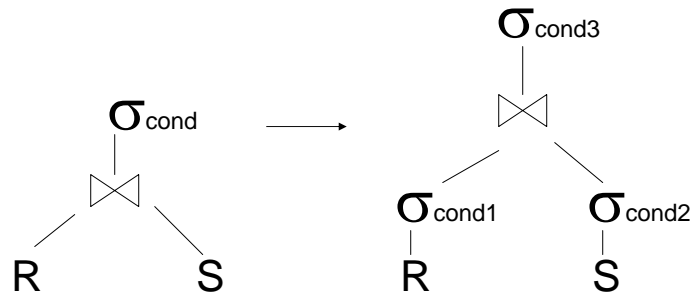
E.g.: in conditions:

$$(S.A=1) \wedge (S.A>5) \Rightarrow \text{False}$$

$$(S.A<10) \wedge (S.A<5) \Rightarrow S.A<5$$

Algebraic rewriting

E.g.: Push conditions down



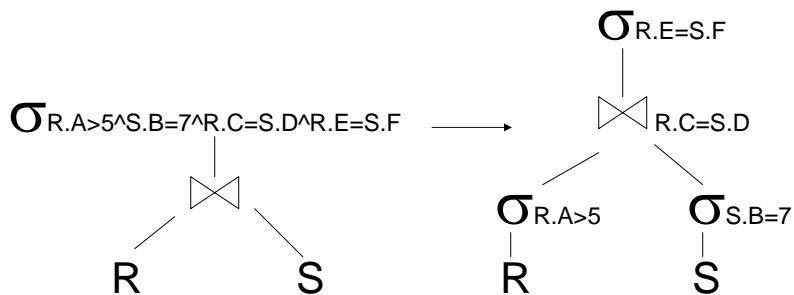
CS5225

Query Processing

9

Algebraic rewriting

E.g.: Push conditions down



CS5225

Query Processing

10

- After decomposition:
 - algebraic query tree on relations
- Localization:
 - Replace relations by corresponding fragments

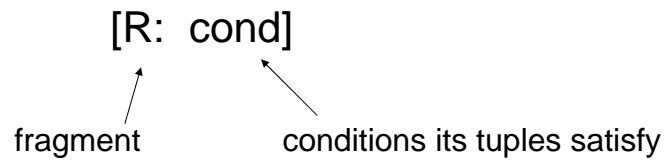
Localization steps

- (1) Start with query
- (2) Replace relations by fragments
- (3) Push \cup : up (use CS3223 rules)

π, σ : down

- (4) Simplify – eliminate unnecessary operations

Notation for fragment



CS5225

Query Processing

13

Example A

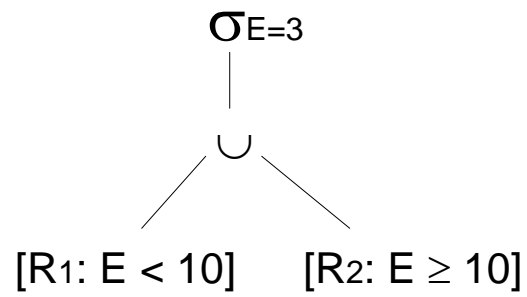
(1) $\sigma_{E=3}$
 |
 R

CS5225

Query Processing

14

(2)

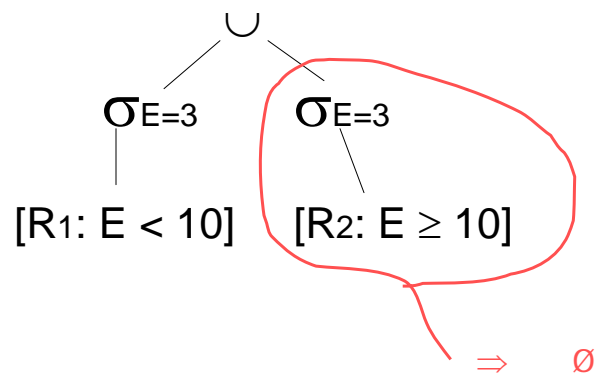


CS5225

Query Processing

15

(3)



CS5225

Query Processing

16

(4)

$$\begin{array}{c} \sigma_{E=3} \\ | \\ [R_1: E < 10] \end{array}$$

CS5225

Query Processing

17

Rule 1

Ⓐ $\sigma_{c_1}[R: c_2] \Rightarrow [\sigma_{c_1} R: c_1 \wedge c_2]$

Ⓑ $[R: \text{False}] \Rightarrow \emptyset$

In example A:

$$\sigma_{E=3}[R_2: E \geq 10] \Rightarrow [\sigma_{E=3} R_2: E=3 \wedge E \geq 10]$$

$$\Rightarrow [\sigma_{E=3} R_2: \text{False}]$$

$$\Rightarrow \emptyset$$

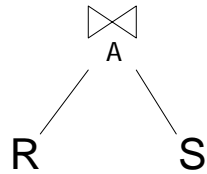
CS5225

Query Processing

18

Example B

(1)



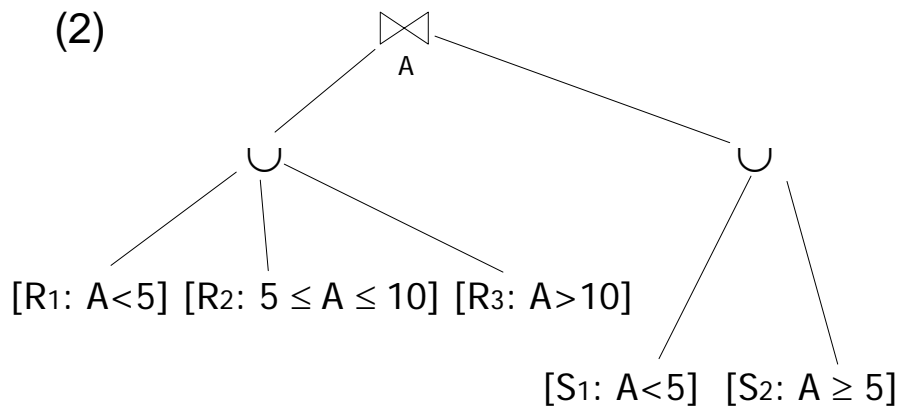
A=common
attribute

CS5225

Query Processing

19

(2)



CS5225

Query Processing

20

Rule 2

$$[R: C_1] \bowtie_A [S: C_2] \Rightarrow$$
$$[R \bowtie_A S: C_1 \wedge C_2 \wedge R.A = S.A]$$

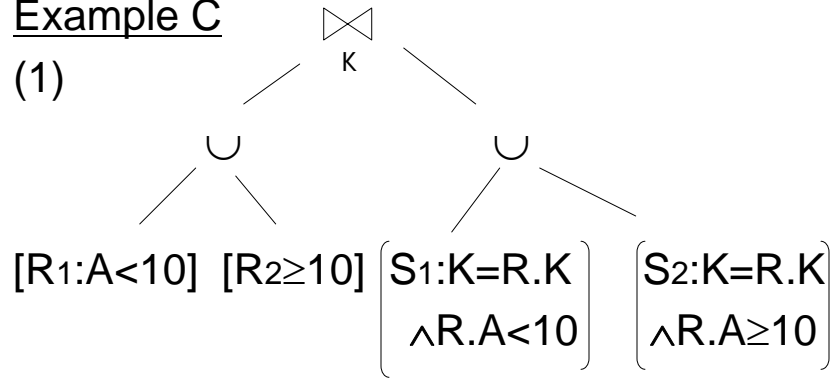
➔ In step 4 of Example B:

$$[R_1: A < 5] \bowtie_A [S_2: A \geq 5]$$
$$\Rightarrow [R_1 \bowtie_A S_2: R_1.A < 5 \wedge S_2.A \geq 5 \wedge R_1.A = S_2.A]$$
$$\Rightarrow [R_1 \bowtie_A S_2: \text{False}] \Rightarrow \emptyset$$

Localization with derived fragmentation

Example C

(1)

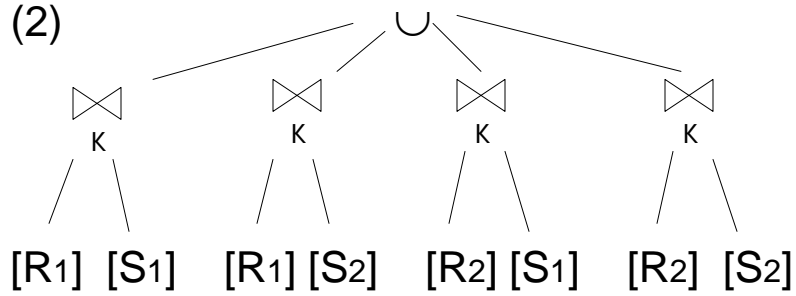


CS5225

Query Processing

25

(2)

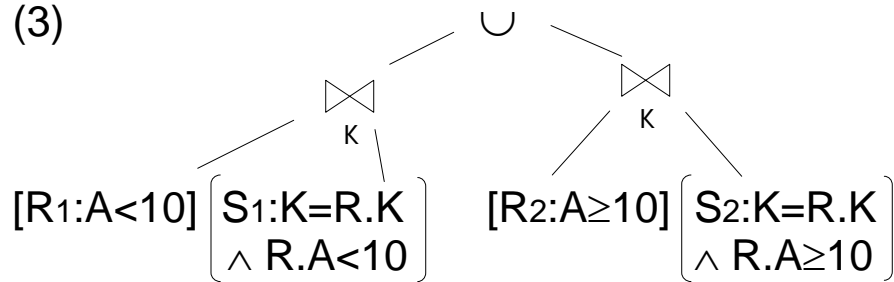


CS5225

Query Processing

26

(3)



CS5225

Query Processing

27

➔ In step 3 of Example C:

$[R1:A<10] \underset{K}{\bowtie} [S2:K=R.K \wedge R.A \geq 10]$

$\Rightarrow [R1 \underset{K}{\bowtie} S2: R1.A < 10 \wedge S2.K = R.K \wedge R.A \geq 10 \wedge R1.K = S2.K]$

$\Rightarrow [R1 \underset{K}{\bowtie} S2: \text{False}] \quad (\text{K is key of R, R1})$

$\Rightarrow \emptyset$

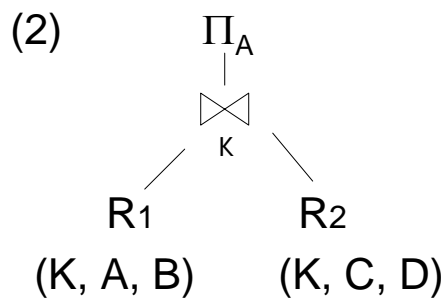
CS5225

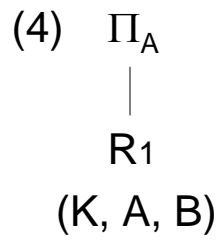
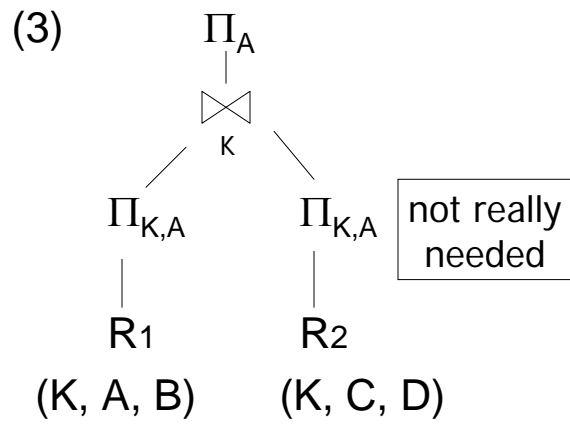
Query Processing

28

- Localization with vertical fragmentation

Example D





Rule 3

- Given vertical fragmentation of R:

$$R_i = \Pi_{A_i}(R), \quad A_i \subseteq A$$

- Then for any $B \subseteq A$:

$$\Pi_B(R) = \Pi_B \left[\bigotimes_i R_i \mid B \cap A_i \neq \emptyset \right]$$

- Localization with hybrid fragmentation

Example E

$$R_1 = \sigma_{k < 5} [\Pi_{k,A} R]$$

$$R_2 = \sigma_{k \geq 5} [\Pi_{k,A} R]$$

$$R_3 = \Pi_{k,B} R$$

Query: Π_A
|
 $\sigma_{k=3}$
|
R

Reduced
Query: Π_A
|
 $\sigma_{k=3}$
|
R1

Summary - Query Processing

- Decomposition ✓
- Localization ✓
- Optimization
 - Tricks for joins
 - Strategies for optimization

CS5225

Query Processing

37

Semi-join

- Goal: reduce communication traffic

$$\bullet R \bowtie_A S \Rightarrow (R \bowtie_A S) \bowtie_A S \text{ or}$$

$$R \bowtie_A (S \bowtie_A R) \text{ or}$$

$$(R \bowtie_A S) \bowtie_A (S \bowtie_A R)$$

$$R \bowtie_A S = \Pi_R (R \bowtie_A S)$$

CS5225

Query Processing

38

	A	B		A	C
R	2	a	S	3	x
	10	b		10	y
	25	c		15	z
	30	d		25	w
				32	x

Computing transmitted data in example:

- with join $R \bowtie S$:
- Ship R to S (Why?); Evaluate join at S Site;
Ship result to query site (assume 3rd site)
- $T = 4 |A+B| + \text{result}$

CS5225

Query Processing

39

Example: $R \bowtie S$

	A	B		A	C
R	2	a	S	3	x
	10	b		10	y
	25	c		15	z
	30	d		25	w
				32	x

Ans: $R \bowtie S$

$\Pi_A R = [2, 10, 25, 30]$

$S \bowtie R =$

A	C
10	y
25	w

CS5225

Query Processing

40

	A	B		A	C
R	2	a	S	3	x
	10	b		10	y
	25	c		15	z
	30	d		25	w
				32	x

Computing transmitted data in example:

- with semi-join $R \bowtie (S \times R)$:

$$T = 4 |A| + 2 |A+C| + \text{result}$$

- with join $R \bowtie S$:

$$T = 4 |A+B| + \text{result}$$

better if say
|B| is large

In general:

- Say R is smaller relation
- $(R \bowtie_A S) \bowtie_A S$ better than $R \bowtie_A S$ if

$$\text{size}(\Pi_A S) + \text{size}(R \bowtie_A S) < \text{size}(R)$$

- Similar comparisons for other semi-joins
- Remember: only taking into account transmission cost

Three-way joins with semi-joins

Goal: $R \bowtie S \bowtie T$

Option 1: $R' \bowtie S' \bowtie T$

where $R' = R \bowtie S$; $S' = S \bowtie T$

Option 2: $R'' \bowtie S' \bowtie T$

where $R'' = R \bowtie S'$; $S' = S \bowtie T$

⇔ Many options!

⇔ Number of semi-join options is exponential in # of relations in join

2-way Semijoin

2-way Semijoin – an extended version of the semijoin

- *Definition*: A 2-way semijoin (t) of R_i and R_j on attribute A can be denoted as

$$R_i \leftarrow A \rightarrow R_j = \{R_i.A \rightarrow R_j, R_j.A \rightarrow R_i\}$$

So t reduces R_i and R_j to R_i' and R_j' respectively.

2-way Semijoin

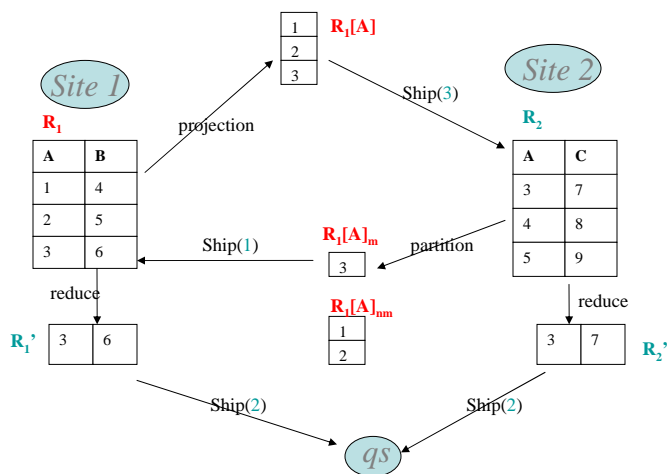
- *Computing steps:*
 - 1) Send $R_i[A]$ from site i to site j ;
 - 2) Reduce R_j to R_j' by eliminating tuples whose attribute A are not matching any of $R_i[A]$ and at the same time partition $R_i[A]$ to $R_i[A]_m$ (match one of $R_j[A]$) and $R_i[A]_{nm}$ ($R_i[A] - R_i[A]_m$);
 - 3) Send $\min(R_i[A]_m, R_i[A]_{nm})$ back to site i ;
 - 4) Reduce R_i to R_i' using $R_i[A]_m$ (or $R_i[A]_{nm}$).
 - 5) Send to R_i' and R_j' join site (3rd site)
- *Evaluation:*
 - $S(R)$ denote size of R
 - Benefit: $B(t) = [S(R_i) - S(R_i')] + [S(R_j) - S(R_j')]$
 - Cost: $C(t) = S(R_i[A]) + \min[S(R_i[A]_m), S(R_i[A]_{nm})]$
 - If the benefit exceeds the cost, then it is called a *cost-effective* 2-way semijoin

CS5225

Query Processing

45

2-way Semijoin



CS5225

Query Processing

46

Trick: Encode $\Pi_A R$ as a bit vector

	A	B
R	2	a
	10	b
	25	c
	30	d

	A	C
S	3	x
	10	y
	15	z
	25	w
	32	x

Ans:
 $R \bowtie S$

$\text{hash}(\Pi_A R) = [001000000001\dots]$

$S \bowtie R =$

	A	C
	10	y
	25	w

Pros/Cons?