CS4224/CS5424 Lecture 10
Distributed Query Optimization

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

Why Optimize?

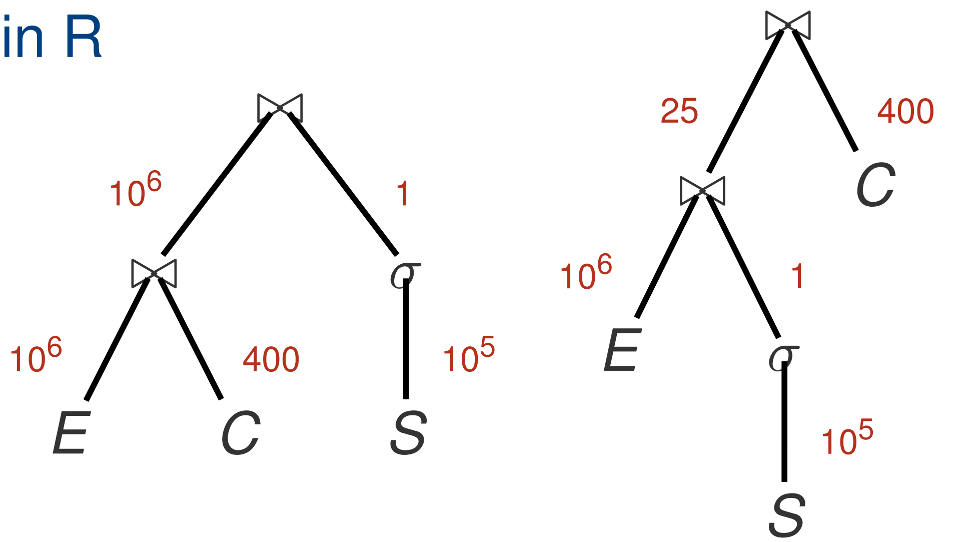
Student (<u>sid</u>, sname, major)
Course (<u>cid</u>, cname, area)
Enrol (<u>sid,cid</u>, grade)

```
SELECT   *
FROM     Student S, Course C, Enrol E
WHERE    E.sid = S.sid
AND      E.cid = C.cid
AND      S.sid = 123
```

Example Query Plans:

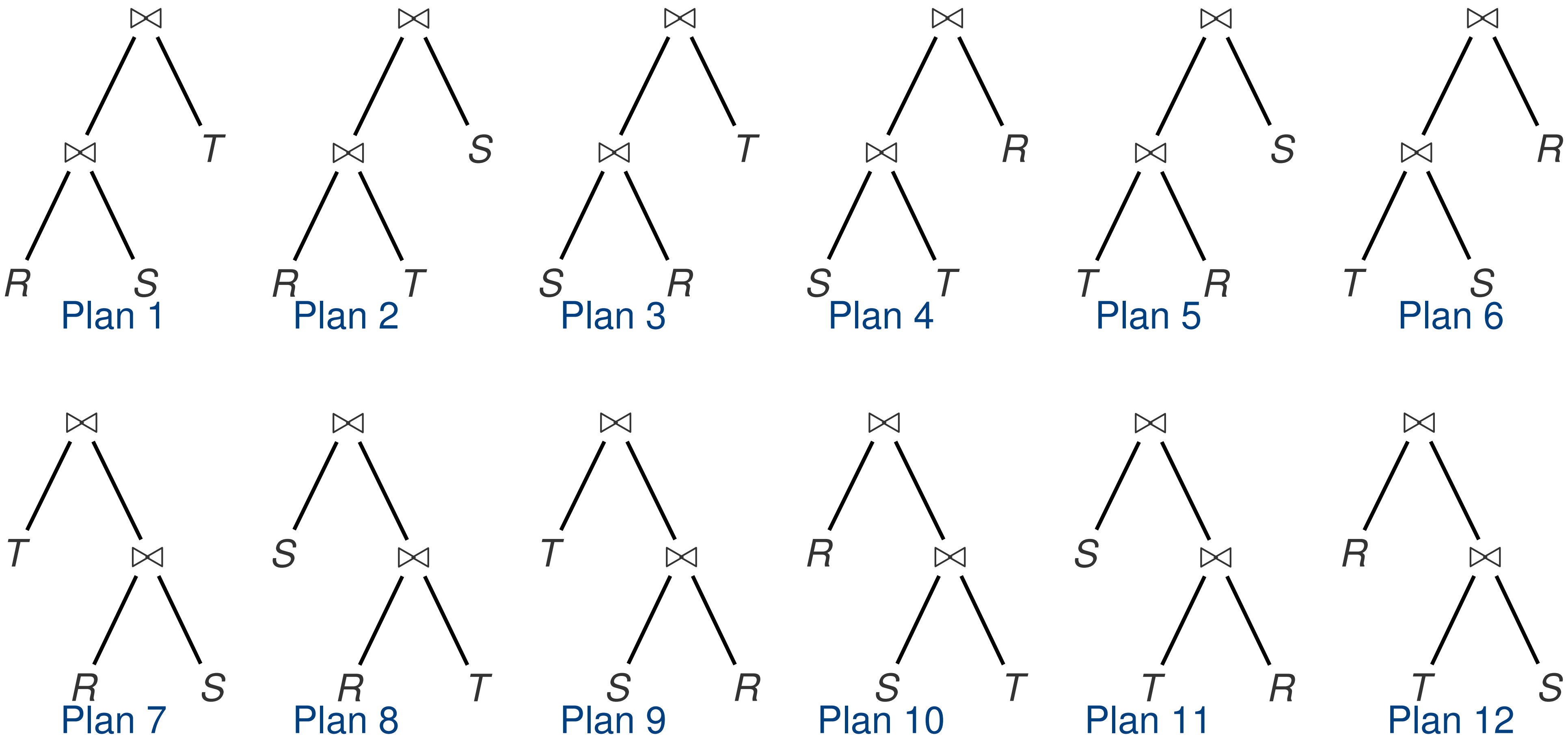
card(*R*) denote the number of tuples in R

$card(C) = 400$
$card(E) = 10^6$
$card(S) = 10^5$
$card(\sigma_{sid=123}(S)) = 1$
$card(C \bowtie_{cid} E) = 10^6$
$card(E \bowtie_{sid} \sigma_{sid=123}(S)) = 25$



Plan 1 Plan 2

Query Plan Search Space for $R \bowtie S \bowtie T$

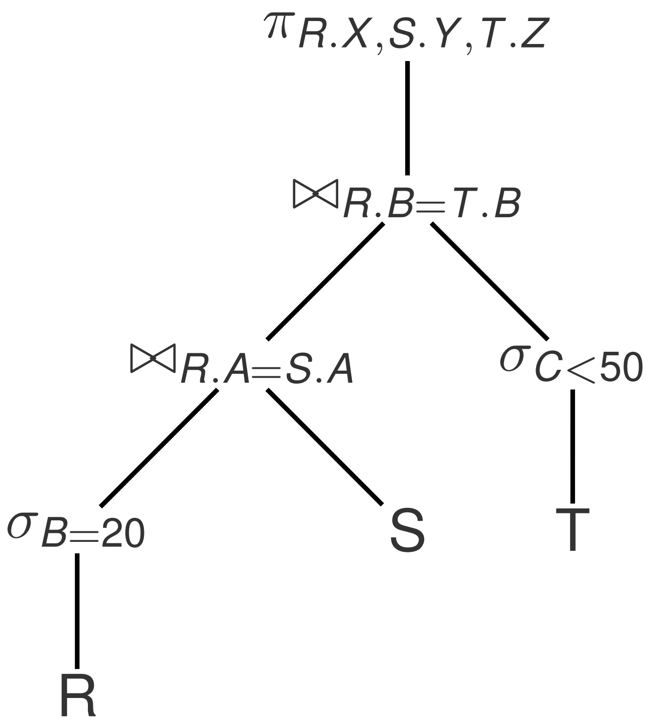


Plan 1 Plan 2 Plan 3 Plan 4 Plan 5 Plan 6

Plan 7 Plan 8 Plan 9 Plan 10 Plan 11 Plan 12

Cost Estimation of Query Plans

1. What is the evaluation cost of each operation?
   - Depends on: size of input operands, available buffer pages, available indexes, etc.
2. What is the output size of each operation?



Cost Estimation of Query Plans (cont.)

Cost model consists of the following:

- Cost model for each operator's algorithms
- Estimation assumptions
  - **Uniformity assumption**: uniform distribution of attribute values
  - **Independence assumption**: independent distribution of values in different attributes
  - **Inclusion assumption**: inclusion dependency between join columns
- Database statistics

Database Statistics

- length(A) = size of attribute A (in bytes)
- length(R) = size of tuple in relation R (in bytes)
- card(R) = number of tuples in relation R
- size(R) = size of relation R (in bytes)
  - size(R) = card(R) $\times$ length(R)
- $card(\pi_A(R))$ = number of distinct values of attribute R.A
- $\min(\pi_A(R))$ = minimum value of attribute R.A
- $\max(\pi_A(R))$ = maximum value of attribute R.A

Selectivity Factors

- $SF(op)$ = selectivity factor of operation op
  - Proportion of tuples of operand relation that participate in result of operation

$$SF(\sigma_p(R)) = \frac{card(\sigma_p(R))}{card(R)}$$

$$SF(R \bowtie S) = \frac{card(R \bowtie S)}{card(R) \times card(S)}$$

$$SF(R \ltimes_A S) = \frac{card(R \ltimes_A S)}{card(R)}$$

Selectivity Factors (cont.)

- Selectivity factors are used to estimate the cardinality of final/intermediate results
- $card(\sigma_p(R)) = SF(\sigma_p(R)) \times card(R)$
- $card(R \bowtie S) = SF(R \bowtie S) \times card(R) \times card(S)$
- $card(R \ltimes_A S) = SF(R \ltimes_A S) \times card(R)$

Estimation of Selectivity Factors

$$
\begin{aligned}
SF(\sigma_{A=v}(R)) &\approx \frac{1}{card(\pi_A(R))} \\
SF(\sigma_{A<v}(R)) &\approx \frac{v-\min(\pi_A(R))}{\max(\pi_A(R))-\min(\pi_A(R))+1} \\
SF(\sigma_{p_1 \wedge p_2}(R)) &\approx SF(\sigma_{p_1}(R)) \times SF(\sigma_{p_2}(R))
\end{aligned}
$$

Estimation of Selectivity Factors

- Inclusion assumption: Consider $R \bowtie_A S$

  $$\text{If } card(\pi_A(R)) \leq card(\pi_A(S)), \text{ then}$$
  $$\pi_A(R) \subseteq \pi_A(S)$$

- Join selectivity estimation:

  $$\boxed{SF(R \bowtie_A S) \approx \frac{1}{\max\{card(\pi_A(R)),\ card(\pi_A(S))\}}}$$

- **Example**: Consider query Q: $R \bowtie_{dept} S$

R

name	dept
Alice	CS
Bob	CS
Carol	CS
Dave	CS
Eve	CS
Fred	CS
George	CS
Henry	EE
Ivy	EE
Jane	EE

S

dept	course
CS	CS101
CS	CS111
CS	CS302
Maths	MA105
Maths	MA203
Music	MU108
Physics	PH113
Physics	PH203

$$card(R) = 10, \quad card(\pi_{dept}(R)) = 2$$
$$card(S) = 8, \quad card(\pi_{dept}(S)) = 4$$

$$card(Q) \approx card(R) \times \frac{card(S)}{card(\pi_{dept}(S))}$$
$$= 10 \times \frac{8}{4} = 20$$

Estimation of Selectivity Factors (cont.)

$$SF(R \ltimes_A S) \approx SF_{SJ}(S.A)$$

$SF_{SJ}(S.A)$ = semijoin selectivity factor of $S.A$

$$SF_{SJ}(S.A) = \frac{card(\pi_A(S))}{|domain(A)|}$$

Distributed Query Optimization

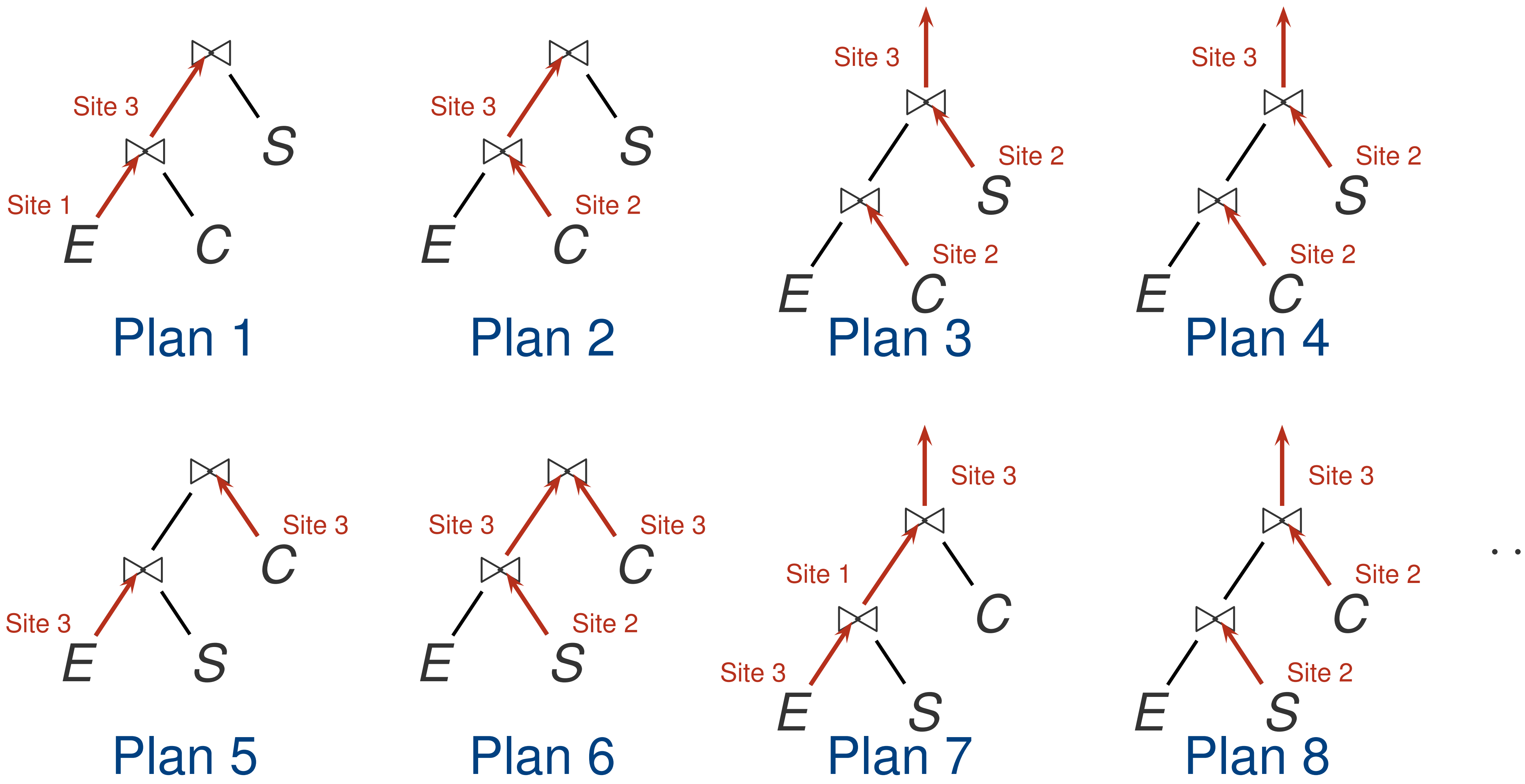
- Additional complexities
  - Data fragmentation & allocation
  - Communication cost
- Optimization techniques to reduce communication cost
  - Semijoin reductions

Search Space: Example

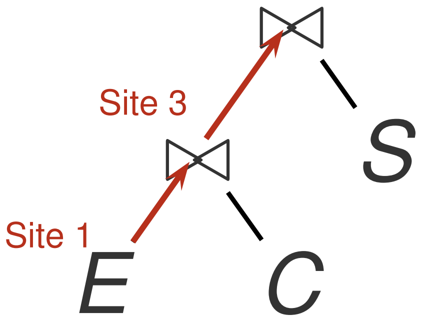
- **Site 1**: `Course` (<u>cid</u>, cname, area)
- **Site 2**: `Enrol` (<u>sid,cid</u>, grade)
- **Site 3**: `Student` (<u>sid</u>, sname, major)
- **Query at Site 3**:

```
SELECT   *
FROM     Student S, Course C, Enrol E
WHERE    E.sid = S.sid
AND      E.cid = C.cid
```

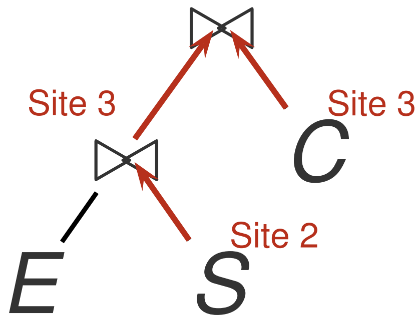
Search Space: Example (cont.)



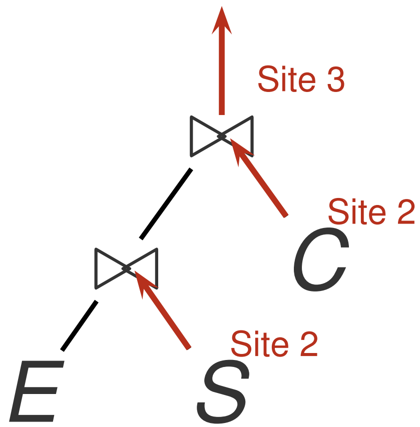
Query Plan Notation



$$(E \overset{1}{\bowtie} C) \overset{3}{\bowtie} S$$



$$(E \overset{2}{\bowtie} S) \overset{3}{\bowtie} C$$



$(E \overset{2}{\bowtie} S) \overset{2}{\bowtie} C$ with result sent to Site 3

Optimizing Total Cost

- Total cost = Sum of CPU, I/O & communication costs.
- CPU cost = $T_{CPU} \times$ (number of CPU instructions)
  - T_{CPU} = time of a CPU instruction
- I/O cost = $T_{I/O} \times$ (number of disk I/Os)
  - $T_{I/O}$ = time of a disk I/O
- Communication cost = $T_{MSG} \times$ (number of messages) $+$ $T_{TR} \times$ (size of transferred data)
  - T_{MSG} = fixed overhead for each message transmission
  - T_{TR} = time to transmit one data unit

Optimization with Semijoins

- $R \ltimes_A S = \pi_{attributes(R)}(R \bowtie_A S) = R \bowtie_A \pi_A(S)$

$$
\begin{aligned}
R \bowtie_A S &= (R \ltimes_A S) \bowtie_A S \\
&= (R \bowtie_A \pi_A(S)) \bowtie_A S
\end{aligned}
$$

- A tuple $t \in R$ is a dangling tuple wrt $R \bowtie S$ if t does not join with any tuple in S
  - i.e., $t \notin \pi_{attributes(R)}(R \bowtie S)$
- $R \ltimes_A S$ eliminates dangling tuples in R (wrt $R \bowtie_A S$)

Optimization with Semijoins: Example

Student

sid	name	major	year
1	Charlie	CS	2
2	Franklin	Maths	4
3	Lucy	Maths	3
4	Marcie	Music	2
5	Patty	Physics	4
6	Sally	CS	3

Project

pid	title	abstract	advisor	sid
1	...	...	...	5
2	...	...	...	2
3	...	...	...	3

Student $\bowtie_{sid}$ Project

sid	name	major	year	pid	title	abstract	advisor
2	Franklin	Maths	4	2	...	...	...
3	Lucy	Maths	3	3	...	...	...
5	Patty	Physics	4	1	...	...	...

Optimization with Semijoins: Example

Student

sid	name	major	year
1	Charlie	CS	2
2	Franklin	Maths	4
3	Lucy	Maths	3
4	Marcie	Music	2
5	Patty	Physics	4
6	Sally	CS	3

Project

pid	title	abstract	advisor	sid
1	...	...	...	5
2	...	...	...	2
3	...	...	...	3

Student $\ltimes_{sid}$ Project

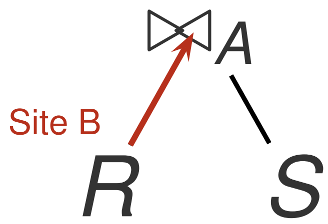
sid	name	major	year
2	Franklin	Maths	4
3	Lucy	Maths	3
5	Patty	Physics	4

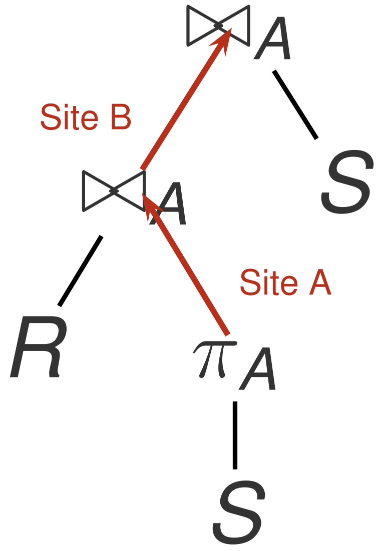
(Student $\ltimes_{sid}$ Project) $\bowtie_{sid}$ Project

sid	name	major	year	pid	title	abstract	advisor
2	Franklin	Maths	4	2	...	...	...
3	Lucy	Maths	3	3	...	...	...
5	Patty	Physics	4	1	...	...	...

Optimization with Semijoins

- **Example**: Site A: R, Site B: S, $size(R) < size(S)$





Direct Join Plan

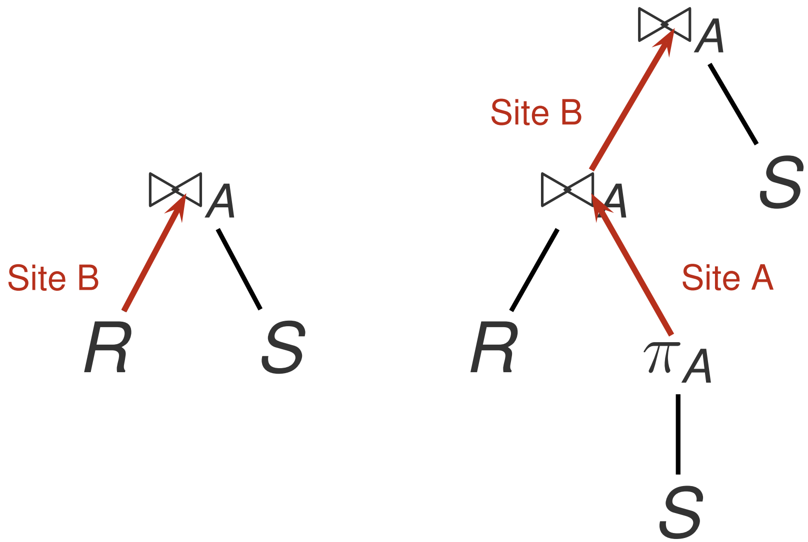
$R \bowtie_A S$

Semijoin Plan

$(R \ltimes_A S) \bowtie_A S$

Optimization with Semijoins (cont.)

- **Direct-join plan**:
  - Sends R over to site of S
  - Joins R & S at site of S
- **Semijoin plan**:
  - Sends $\pi_A(S)$ to site of R
  - Joins R & $\pi_A(S)$ to eliminate dangling tuples in R
  - Sends non-dangling tuples of R (i.e. $R \ltimes_A S$) to site of S
  - Joins R & S at site of S

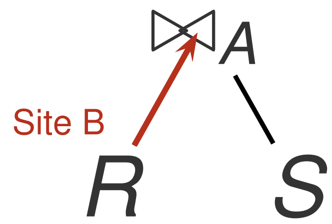


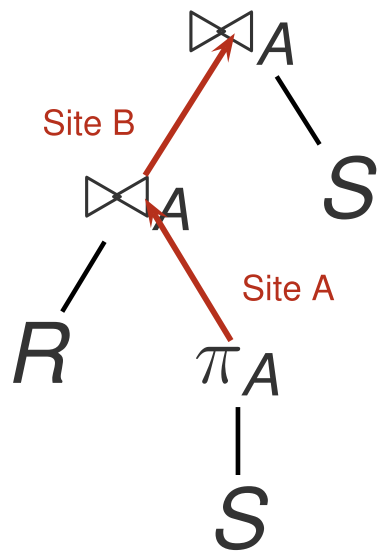
Beneficial Semijoins

- The cost & benefit of applying semi-join optimization is defined relative to the direct-join plan
  - **Cost** measures the additional communication overhead incurred by semijoin plan over direct-join plan
  - **Benefit** measures the communication savings from semijoin plan over direct-join plan
- **Cost($R \ltimes_A S$)** = cost of sending $\pi_A(S)$
- **Benefit($R \ltimes_A S$)** = savings in not sending dangling tuples of R wrt $R \bowtie_A S$
  - $SF(R \ltimes_A S)$ = proportion of tuples in R that join with S
  - $1 - SF(R \ltimes_A S)$ = proportion of dangling tuples in R wrt $R \bowtie_A S$
- A semijoin is beneficial if its benefit exceeds its cost

Beneficial Semijoins (cont.)

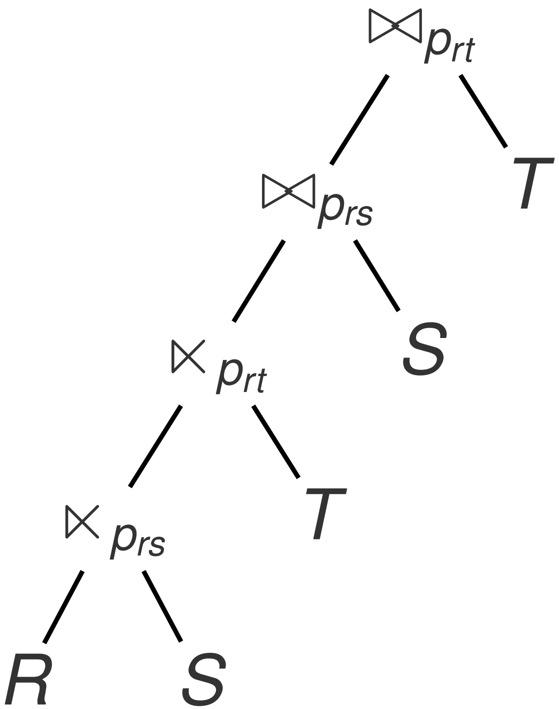
- Assume $size(R) < size(S)$
- Cost($R \ltimes_A S$) = $T_{MSG} + T_{TR} \times size(\pi_A(S))$
- Benefit($R \ltimes_A S$) $= T_{TR} \times size(R) \times (1 - SF(R \ltimes_A S))$
- $R \ltimes_A S$ is a beneficial semijoin if
  Benefit($R \ltimes_A S$) $>$ Cost($R \ltimes_A S$)



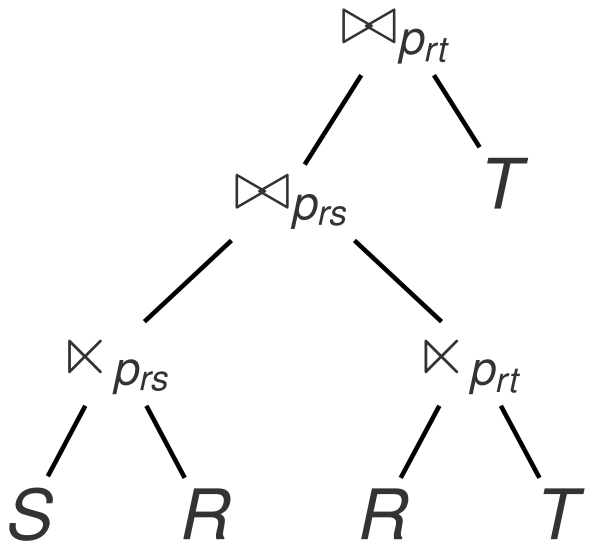


Query Plans with Semijoins

- Larger query plan search space!
- Example: `select * from` R, S, T `where` R.a = S.a `and` R.b = T.b



Query Plan 1



Query Plan 2

Comparable Query Plans

- Two query plans are comparable if they satisfy all the following conditions:
  (1) both plans have the same output schema,
  (2) both plans execute their final operator on the same server, and
  (3) the outputs of both plans are either (a) unordered or (b) sorted in the same order
- Two query plans are incomparable if they are not comparable
- Examples
  - $R \overset{1}{\bowtie}_p S$ and $R \overset{1}{\ltimes}_p S$ are incomparable
  - $R \overset{1}{\bowtie}_p S$ and $R \overset{2}{\bowtie}_p S$ are incomparable
  - $(R \overset{1}{\ltimes}_p S) \overset{2}{\bowtie}_p (S \overset{3}{\ltimes}_p R)$ and $R \overset{2}{\bowtie}_p S$ are comparable if they satisfy condition (3)

Query Plans

- Given a query plan P, let cost(P) denote the cost of executing P
- Given two query plans, P1 & P2, P1 is better than P2 (or P2 is worse than P1) if (1) P1 & P2 are comparable and (2) $cost(P1) < cost(P2)$
- Consider a query Q over a set of relations R, and $S \subseteq R$, $S \neq \emptyset$. Let optPlan(S) denote the set of **optimal query subplans** of Q over S
  - For every plan $P \in optPlan(S)$, there does not exist another plan P' that is better than P
- Let LOptPlan(S) denote the logical query plans in optPlan(S)

Classic Dynamic Programming (DP) Algorithm (Stocker, et al., ICDE 2001)

Classic_DP (Q)
Input: A SPJ query Q on relations $R_1, \cdots, R_n$
Output: A query plan for Q

for i = 1 to n do {
    optPlan($\{R_i\}$) = accessPlans(R_i)
    prunePlans(optPlan($\{R_i\}$))
}
for i = 2 to n do
    for all $S \subseteq \{R_1, \cdots, R_n\}$ such that $|S| = i$ do {
        optPlan(S) = $\emptyset$
        for all $O \subset S$ such that $O \neq \emptyset$ do {
            optPlan(S) = optPlan(S) $\cup$ joinPlans(optPlan(O), optPlan(($S - O$)))
            prunePlans(optPlan(S))
        }
    }
return optPlan($\{R_1, \cdots, R_n\}$)

Query Plan Enumeration: Example

- Distributed Database:
  - Site 1: **R**(A,B,C,D)
  - Site 2: **S**(X,Y)
  - Site 3: **T**(E,F,G)
- Query submitted at Site 1:

select *
from R **join** S **on** R.A = S.X **join** T on R.D = T.F
where R.B $>$ 10
and R.C = 20
and T.E < 100

- Available indexes: I_B, I_C, I_E
- Assumptions on database system
  - Supports only one join algorithm: hash join
  - Avoids cartesian products

Example: Single-relation Plans

$$\sigma_p(R \bowtie_{R.A=S.X} S \bowtie_{R.D=T.F} T),\ p = (R.B > 10) \wedge (R.C = 20) \wedge (T.E < 100)$$

- **Plans for $\{$R$\}$**
  - Plan P1: Table scan with "$(B > 10) \wedge (C = 20)$"
  - Plan P2: Index seek with I_B & RID-lookups with "$C = 20$"
  - Plan P3: Index seek with I_C & RID-lookups with "$B > 10$"
  - Plan P4: Index intersection with I_B & I_C, and RID-lookups
  - Before pruning: optPlan($\{R\}$) = $\{P1, P2, P3, P4\}$
  - Assume $cost(P3) < cost(P4) < cost(P2) < cost(P1)$
  - After pruning: optPlan($\{R\}$) = $\{P3\}$
- **Plans for $\{$S$\}$**
  - Plan P5: Table scan of S
  - optPlan($\{S\}$) = $\{P5\}$
- **Plans for $\{$T$\}$**
  - Plan P6: Table scan of T with "$(E < 100)$"
  - Plan P7: Index seek with I_E & RID-lookups
  - Before pruning: optPlan($\{T\}$) = $\{P6, P7\}$
  - Assume $cost(P7) < cost(P6)$
  - After pruning: optPlan($\{T\}$) = $\{P7\}$

Example: Two-relation Plans

- **Plans for $\{R, S\}$**
  - Plan P8: optPlan($\{R\}$) $\overset{1}{\bowtie}$ optPlan($\{S\}$) = $P3 \overset{1}{\bowtie} P5$
  - Plan P9: optPlan($\{R\}$) $\overset{2}{\bowtie}$ optPlan($\{S\}$) = $P3 \overset{2}{\bowtie} P5$
  - Plan P10: optPlan($\{R\}$) $\overset{3}{\bowtie}$ optPlan($\{S\}$) = $P3 \overset{3}{\bowtie} P5$
  - Plan P11: optPlan($\{S\}$) $\overset{1}{\bowtie}$ optPlan($\{R\}$) = $P5 \overset{1}{\bowtie} P3$
  - Plan P12: optPlan($\{S\}$) $\overset{2}{\bowtie}$ optPlan($\{R\}$) = $P5 \overset{2}{\bowtie} P3$
  - Plan P13: optPlan($\{S\}$) $\overset{3}{\bowtie}$ optPlan($\{R\}$) = $P5 \overset{3}{\bowtie} P3$
  - Assume after pruning, optPlan($\{R, S\}$) = $\{P8, P9, P10\}$
- **Plans for $\{R, T\}$**
  - Plan P12: optPlan($\{R\}$) $\overset{1}{\bowtie}$ optPlan($\{T\}$) = $P3 \overset{1}{\bowtie} P7$
  - Plan P13: optPlan($\{R\}$) $\overset{3}{\bowtie}$ optPlan($\{T\}$) = $P3 \overset{3}{\bowtie} P7$
  - Plan P14: optPlan($\{R\}$) $\overset{2}{\bowtie}$ optPlan($\{T\}$) = $P3 \overset{2}{\bowtie} P7$
  - Plan P15: optPlan($\{T\}$) $\overset{1}{\bowtie}$ optPlan($\{R\}$) = $P7 \overset{1}{\bowtie} P3$
  - Plan P16: optPlan($\{T\}$) $\overset{3}{\bowtie}$ optPlan($\{R\}$) = $P7 \overset{1}{\bowtie} P3$
  - Plan P17: optPlan($\{T\}$) $\overset{2}{\bowtie}$ optPlan($\{R\}$) = $P7 \overset{2}{\bowtie} P3$
  - Assume after pruning, optPlan($\{R, T\}$) = $\{P12, P13, P17\}$

Example: Three-relation Plans

- **Plans for $\{\mathbf{R}, \mathbf{S}, \mathbf{T}\}$**
  - Plan P18: optPlan($\{R, S\}$) $\overset{1}{\bowtie}$ optPlan($\{T\}$)
    - Plan P18-8: $P8 \overset{1}{\bowtie} P7$
    - Plan P18-9: $P9 \overset{1}{\bowtie} P7$
    - Plan P18-10: $P10 \overset{1}{\bowtie} P7$
  - Plan P19: optPlan($\{R, S\}$) $\overset{2}{\bowtie}$ optPlan($\{T\}$)
  - Plan P20: optPlan($\{R, S\}$) $\overset{3}{\bowtie}$ optPlan($\{T\}$)
  - Plan P21: optPlan($\{T\}$) $\overset{1}{\bowtie}$ optPlan($\{R, S\}$)
  - Plan P22: optPlan($\{T\}$) $\overset{2}{\bowtie}$ optPlan($\{R, S\}$)
  - Plan P23: optPlan($\{T\}$) $\overset{3}{\bowtie}$ optPlan($\{R, S\}$)
  - Plan P24: optPlan($\{R, T\}$) $\overset{1}{\bowtie}$ optPlan($\{S\}$)
  - Plan P25: optPlan($\{R, T\}$) $\overset{2}{\bowtie}$ optPlan($\{S\}$)
  - Plan P26: optPlan($\{R, T\}$) $\overset{3}{\bowtie}$ optPlan($\{S\}$)
  - Plan P27: optPlan($\{S\}$) $\overset{1}{\bowtie}$ optPlan($\{R, T\}$)
  - Plan P28: optPlan($\{S\}$) $\overset{2}{\bowtie}$ optPlan($\{R, T\}$)
  - Plan P29: optPlan($\{S\}$) $\overset{3}{\bowtie}$ optPlan($\{R, T\}$)
  - Assume after pruning, optPlan($\{R, S, T\}$) = $\{$P18-8, P28-17, P23-10$\}$
  - Optimal plan for query is the lowest cost plan in optPlan($\{R, S, T\}$)
    - Plans P28-17 & P23-10 require additional communication cost to send result to Site 1

Enhanced DP Algorithm (Stocker, et al., ICDE 2001)

Enhanced_DP (Q)
Input: A SPJ query Q on relations $R_1, \cdots, R_n$
Output: A query plan for Q

```
for i = 1 to n do {
    optPlan({R_i}) = accessPlans(R_i)
    prunePlans(optPlan({R_i}))
}
for i = 2 to n do
    for all S ⊆ {R_1, ··· , R_n} such that |S| = i do {
        optPlan(S) = ∅
        for all O ⊂ S such that O ≠ ∅ do {
            for all P ⊂ O do {
                optPlan(S) = optPlan(S) ∪ joinPlans(optPlan(O), optPlan((S − O) ∪ P), 0)
                optPlan(S) = optPlan(S) ∪ SJjoinPlans(optPlan(O), optPlan((S − O) ∪ P), 0)
                prunePlans(optPlan(S))
            }
        }
        timestamp = 0
        do {
            Δ = 'new plans with latest timestamp in S'
            for all O ⊆ S such that O ≠ ∅ do {
                optPlan(S) = optPlan(S) ∪ joinPlans(Δ, optPlan(O), timestamp+1)
                optPlan(S) = optPlan(S) ∪ SJjoinPlans(Δ, optPlan(O), timestamp+1)
                prunePlans(optPlan(S))
            }
            timestamp ++
        } while (Δ ≠ ∅)
    }
return optPlan({R_1, ··· , R_n})
```

Incorporating Semijoins

- **Enumerator Extension**
  - Left & right operands may not be disjoint
  - Example: $(R \ltimes S) \ltimes (T \ltimes S)$
  - Algorithm Enhanced DP: Steps N1, N2 & N3
- **Avoiding Redundant Joins & Semijoins**
  - Examples: $R \bowtie S \bowtie S$, $R \ltimes S \ltimes S$
  - Algorithm Enhanced DP: joinPlans() & SJjoinPlans()
- **Fix-point Iteration**
  - Some plans in optPlan(S) may not be complete (i.e, the plan's output schema is not from a join over S without any semijoin)
  - Example: The plan $R \ltimes S$ in optPlan($\{R, S\}$) is not complete
  - Algorithm Enhanced DP: Steps N5 to N14
- **Vertical Pruning**
  - Comparable query plans may be stored across different optPlan() entries
  - Algorithm Enhanced DP: prunePlans()

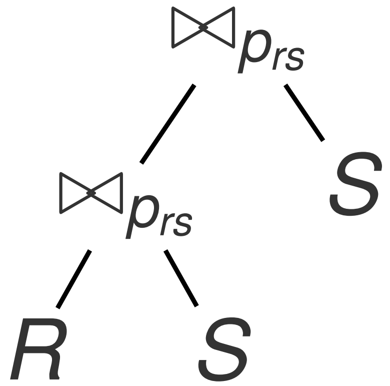
Enumerator Extension

S	O	S-O	P	(S-O) $\cup$ P	LOptPlan(S)
$\{R, S\}$	$\{R\}$	$\{S\}$	$\{\}$	$\{S\}$	$R \bowtie S, R \ltimes S$
$\{R, S\}$	$\{S\}$	$\{R\}$	$\{\}$	$\{R\}$	$S \bowtie R, S \ltimes R$
$\{R, T\}$	$\{R\}$	$\{T\}$	$\{\}$	$\{T\}$	$R \bowtie T, R \ltimes T$
$\{R, T\}$	$\{T\}$	$\{R\}$	$\{\}$	$\{R\}$	$T \bowtie R, T \ltimes R$
$\{R, S, T\}$	$\{R\}$	$\{S, T\}$	$\{\}$	$\{S, T\}$	-
$\{R, S, T\}$	$\{S\}$	$\{R, T\}$	$\{\}$	$\{R, T\}$	$S \bowtie (R \bowtie T)$, $S \ltimes (R \bowtie T)$, $S \bowtie (R \ltimes T)$, $S \ltimes (R \ltimes T)$, $S \bowtie (T \ltimes R)$
$\{R, S, T\}$	$\{T\}$	$\{R, S\}$	$\{\}$	$\{R, S\}$	$T \bowtie (R \bowtie S)$, $T \ltimes (R \bowtie S)$, $T \bowtie (R \ltimes S)$, $T \ltimes (R \ltimes S)$
$\{R, S, T\}$	$\{R, S\}$	$\{T\}$	$\{\}$	$\{T\}$	$(R \bowtie S) \bowtie T$, $(R \bowtie S) \ltimes T$, $(R \ltimes S) \bowtie T$, $(R \ltimes S) \ltimes T$, $(S \bowtie R) \bowtie T$, $(S \bowtie R) \ltimes T$
$\{R, S, T\}$	$\{R, S\}$	$\{T\}$	$\{R\}$	$\{R, T\}$	$(R \bowtie S) \bowtie (R \bowtie T)$, $(R \bowtie S) \bowtie (R \ltimes T)$, $(R \bowtie S) \bowtie (T \bowtie R)$, $(R \bowtie S) \bowtie (T \ltimes R)$, $(R \bowtie S) \ltimes (R \bowtie T)$, $(R \bowtie S) \ltimes (R \ltimes T)$, $(R \bowtie S) \ltimes (T \bowtie R)$, $(R \bowtie S) \ltimes (T \ltimes R)$, ...

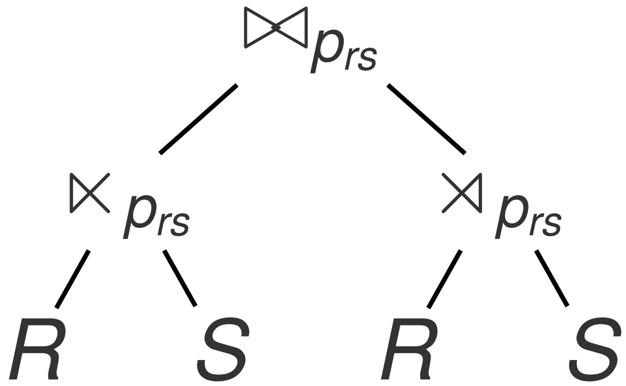
Avoiding Redundant Joins

- Consider the example query $Q = R \bowtie_{p_{rs}} S \bowtie_{p_{rt}} T$
- Each join predicate p_{rs} can appear in three forms in a query plan:
  - Inner-join: $R \bowtie_{p_{rs}} S$
  - left-semijoin: $R \ltimes_{p_{rs}} S$
  - right-semijoin: $R \rtimes_{p_{rs}} S$
- Let **JPred(Q)** denote the set of all forms of join/semijoin predicates in a query plan for Q
- For example query Q, $JPred(Q) = \{ \bowtie_{p_{rs}}, \ltimes_{p_{rs}}, \rtimes_{p_{rs}}, \bowtie_{p_{rt}}, \ltimes_{p_{rt}}, \rtimes_{p_{rt}} \}$
- A query plan P for a quey Q is defined to be a reasonable query plan if for each leaf-to-root path L in P, each predicate $j \in JPred(Q)$ appears at most once in L.
- **joinPlans()** & **SJjoinPlans()** enumerate only reasonable query plans

Avoiding Redundant Joins (cont.)



Query Plan 1



Query Plan 2

Fix-point Iteration

- A query plan P in optPlan(S) is complete if the output schema of P is the same as the output schema from a join over the relations in S (without any semijoin); otherwise it is incomplete

- Consider the plans obtained in optPlan($\{R, S\}$) at the end of step 11 in Enhanced DP Algorithm

- One of the incomplete plans in optPlan($\{R, S\}$) is $R \ltimes S$

- At the end of the first fix-point iteration, the following are two complete plans that are derived from $R \ltimes S$:
  - $(R \ltimes S) \bowtie S$
  - $(R \ltimes S) \bowtie (S \ltimes R)$

Vertical Pruning

- Consider a query Q over the relations $\{R, S, T, U\}$ with join predicates p_{rs}, p_{rt}, p_{ru} & p_{st}
- Consider the query plan $P = R \ltimes_{p_{rs}} (S \ltimes_{p_{st}} T)$ in optPlan($\{R, S, T\}$)
- Query plans that are comparable to P could be found in different optPlan() entries. Example:
  - R in optPlan($\{R\}$)
  - $R \ltimes_{p_{rs}} S$ in optPlan($\{R, S\}$)
  - $R \ltimes_{p_{rt}} T$ in optPlan($\{R, T\}$)
  - $R \ltimes_{p_{ru}} U$ in optPlan($\{R, U\}$)
  - $R \ltimes_{p_{rt}} (T \ltimes_{p_{st}} S)$ in optPlan($\{R, S, T\}$)
  - etc.
- prunePlans: Need to perform both **intra-entry pruning** as well as **inter-entry pruning**

References

- T. Özsu & P. Valdureiz, *Distributed Query Processing*, Chapter 4, Principles of Distributed Database Systems, 4^{th} Edition, 2020
- K. Stocker, et al., *Integrating Semi-Join-Reducers into State-of-the-Art Query Processors*, ICDE 2001