Notes 3: Query Processing
Query Processing

Decomposition
Localization
Optimization
Decomposition

Same as in centralized system

1. Normalization
2. Eliminating redundancy
3. Algebraic rewriting
Normalization

Convert from general language to a *standard* form
E.g., relational algebra
Normalization

Example:

select R.A, R.C
from R, S
where R.A = S.A and
((R.B = 1 and S.D = 2) or
(R.C > 3 and S.D = 2))

conjunctive normal form
Normalization

Also detect invalid expressions

select *
from R  
where R.D = 3  
✓ R does not have D attribute
Eliminating Redundancy

E.g., in conditions

\[(S.A = 1) \land (S.A > 5) \Rightarrow \text{false}\]

\[(S.A < 10) \land (S.A < 5) \Rightarrow S.A < 5\]
Eliminating Redundancy

E.g., sharing common sub-expressions
Algebraic Rewriting

E.g., pushing conditions down

\[
\sigma_{\text{cond}} \bowtie R \bowtie S \\
\sigma_{\text{cond}2} \bowtie \sigma_{\text{cond}3} \\
\sigma_{\text{cond}1} \bowtie R \bowtie S
\]
Query Processing

Decomposition ✔
→ One or more algebraic query trees on relations

Localization
Replacing relations by corresponding fragments
Localization

Steps

(1) Start with query

(2) Replace relations by fragments

(3) Push $\cup$ up and $\prod$, $\sigma$ down (apply CS 245 rules)

(4) Simplify $\sim$ eliminate unnecessary operations
Localization

Notation for fragment

\[ [R : \text{cond}] \]

fragment

conditions its tuples satisfy
Example A

(1)

\[ \sigma_{E=3} \]

| R |
Example A

(2)

\[
\sigma_{E=3} \cup [R_1: E<10] \cup [R_2: E\geq10]
\]
Example A

(3)

\[ \sigma_{E=3} \bigcup \sigma_{E=3} \]

\[
\begin{align*}
\sigma_{E=3} & \quad [R_1: E<10] \\
\sigma_{E=3} & \quad [R_2: E\geq10]
\end{align*}
\]
Example A

(3)

\[ \sigma_{E=3} \cup \begin{cases} R_1: E < 10 \\ R_2: E \geq 10 \end{cases} \implies \emptyset \]
Example A

(4)

\[ \sigma_{E=3} \]

[\[R_1: E<10]\]
Localization

Rule 1

\[ \sigma_{c_1}[R: c_2] \implies \sigma_{c_1}[R: c_1 \land c_2] \]

\[ [R: \text{false}] \implies \emptyset \]
Example A

\[ \sigma_{E=3} \left[ R_2 : E \geq 10 \right] \Rightarrow \sigma_{E=3} \left[ R_2 : E=3 \land E \geq 10 \right] \]
\[ \Rightarrow \sigma_{E=3} \left[ R_2 : \text{false} \right] \]
\[ \Rightarrow \emptyset \]
Example B

(1)

\[
\begin{array}{c}
\bowtie \\
A \\
\Lambda \\
R \\
\Lambda \\
S
\end{array}
\]

A = common attribute
Example B

(2)
(3)

Example B
Example B

(4)

\[ R_1: A < 5 \quad S_1: A < 5 \]

\[ R_2: 5 \leq A \leq 10 \quad S_2: A \geq 5 \]

\[ R_3: A > 10 \quad S_2: A \geq 5 \]
Localization

Rule 2

\[ [R: c_1] \bowtie_A [S: c_2] \implies [R \bowtie_A S: c_1 \land c_2 \land R.A = S.A] \]

\[ [R: \text{false}] \implies \emptyset \]
Example B

\[ R_1: A < 5 \] \bowtie_A \ [S_2: A \geq 5] \]

\[ \Rightarrow [R_1 \bowtie_A S_2: R_1.A < 5 \land S_2.A \geq 5 \land R_1.A=S_2.A] \]

\[ \Rightarrow [R_1 \bowtie_A S_2: \text{false}] \]

\[ \Rightarrow \emptyset \]
Example C

(2)

\[ R_1: A < 10 \]
\[ R_2: A \geq 10 \]
\[ S_1: K = R.K \land R.A < 10 \]
\[ S_2: K = R.K \land R.A \geq 10 \]

derived fragmentation
Example C

(3)

\[ R_1 \bowtie K \bowtie K \bowtie K \bowtie K \cup S_1 \]

\[ [R_1] \bowtie [S_1] \bowtie [R_1] \bowtie [S_2] \bowtie [R_2] \bowtie [S_1] \bowtie [R_2] \bowtie [S_2] \]
Example C

(3)

\[
\left\{ K \right\} \bowtie \left\{ K \right\} \bowtie \left\{ K \right\} \bowtie \left\{ K \right\} \cup \\
\left\{ S_1 \right\} \bowtie \left\{ S_1 \right\} \bowtie \left\{ S_2 \right\} \bowtie \left\{ S_2 \right\}
\]
Example C

(4)

\[ R_1: A < 10 \]
\[ S_1: K = R.K \land R.A < 10 \]
\[ R_2: A \geq 10 \]
\[ S_2: K = R.K \land R.A \geq 10 \]
Example C

\[ R_1: A < 10 \] \bowtie_K \ [S_2: K = R.K \land R.A \geq 10] \]

\[ \Rightarrow [R_1 \bowtie_K S_2: R_1.A < 10 \land S_2.K = R.K \land R.A \geq 10 \land R_1.K = S_2.K] \]

\[ \Rightarrow [R_1 \bowtie_K S_2: \text{false}] \]

\[ \Rightarrow \emptyset \]
Example D

(1) \[ \Pi_A \quad \begin{cases} R1 (K, A, B) \\ R2 (K, C, D) \end{cases} \]

vertical fragmentation
Example D

(2) \[ \prod_A \]

\[ \bowtie K \]

\[ R_1 \quad R_2 \]

(K,A,B) (K,C,D)
Example D

(2) \[ \prod_A \bowtie_K \prod_{K,A} R_1 \bowtie_K \prod_{K,A} R_2 \]

\[ (K,A,B) \bowtie (K,C,D) \]
Example D

(4) \[ \Pi_A \]

\[ \text{R}_1 \]

(K, A, B)
Rule 3

Consider the vertical fragmentation

\[ R_i = \prod_{A_i}(R), A_i \subseteq A \]

Then for any \( B \subseteq A \)

\[ \prod_B(R) = \prod_B(\bowtie_i R_i \mid B \cap A_i \neq \emptyset) \]
Query Processing

Decomposition ✔
Localization ✔

Optimization
Overview
Joins and other operations
Inter-operation parallelism
Optimization strategies
Overview

Optimization process
1. Generate query plans
2. Estimate size of intermediate results
3. Estimate cost of plan
4. Pick minimum

$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_n$

$C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_n$

4. Pick minimum
Overview

Differences from centralized optimization

New strategies for some operations
- Parallel/distributed sort
- Parallel/distributed join
  - Semi-join
  - Privacy preserving join
- Duplicate elimination
- Aggregation
- Selection

Many ways to assign and schedule processors
Parallel/Distributed Sort

Input
a) Relation $R$ on single site/disk
b) Relation $R$ fragmented/partitioned by sort attribute
c) Relation $R$ fragmented/partitioned by another attribute
Parallel/Distributed Sort

Output

a) Sorted $R$ on single site/disk
b) Sorted $R$ fragments/partitions

\[ \begin{array}{c|c}
5 & \ldots \\
6 & \\
10 & \\
\hline
12 & \ldots \\
15 & \\
\hline
19 & \ldots \\
20 & \\
21 & \\
50 & \\
\end{array} \]

$F_1$ $F_2$ $F_3$
Basic Sort

\[ R(K, \ldots) \text{ to be sorted on } K \]
Horizontally fragmented on \( K \) using vector \((k_0, k_1, \ldots, k_n)\)
Basic Sort

Algorithm
1. Sort each fragment independently
2. Ship results if necessary
Basic Sort

Same idea on different architectures

Shared nothing

Shared memory
Range-Partition Sort

\( R(\text{K, ...}) \) to be sorted on \( \text{K} \)

\( R \) located at one or more site/disk, **not** fragmented on \( \text{K} \)
Range-Partition Sort

Algorithm
1. Range partition on $K$
2. Basic sort
Partition Vectors

Problem
Select a good partition vector given fragments

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ R_a \]
\[ R_b \]
\[ R_c \]
Partition Vectors

Example approach

Each site sends to coordinator
- MIN sort key
- MAX sort key
- Number of tuples

Coordinator
- Computes vector and distributes to sites
- Decides the number of sites to perform local sorts
Partition Vectors

Sample scenario
Coordinator receives

\[ S_A: \quad \text{MIN} = 5 \quad \text{MAX} = 9 \quad \# \text{of tuples} = 10 \]

\[ S_B: \quad \text{MIN} = 7 \quad \text{MAX} = 16 \quad \# \text{of tuples} = 10 \]
Sample scenario

Coordinator receives

$S_A$:  MIN = 5  MAX = 9  # of tuples = 10

$S_B$:  MIN = 7  MAX = 16  # of tuples = 10

Expected tuples

\[ k_0 = \frac{10}{16 - 7 + 1} \]

\[ 2 = \frac{10}{9 - 5 + 1} \]

assuming we want to sort at 2 sites
Partition Vectors

Sample scenario

Expected tuples

Expected tuples with key < \( k_0 \) = half of total tuples

\[
2(k_0 - 5) + (k_0 - 7) = 10
\]

\[
k_0 = (10 + 10 + 7) / 3 = 9
\]
Partition Vectors

Variations
Send more info to coordinator:
a) Partition vector for local site

\[\begin{array}{cccccc}
3 & 3 & 3 & \text{# of tuples} \\
5 & 6 & 8 & 10 & \text{local vector}
\end{array}\]

b) Histogram

\[\begin{array}{cccccc}
5 & 6 & 7 & 8 & 9 & 10
\end{array}\]
Partition Vectors

Variations
Multiple rounds

1. Sites send range and # of tuples to coordinator
2. Coordinator distributes preliminary vector $V_0$
3. Sites send coordinator # of tuples in each $V_0$ range
4. Coordinator computes and distributes final vector $V_f$
Partition Vectors

Variations
Distributed algorithm that does not require a coordinator?
Parallel External Sort-Merge

Same as range-partition sort except does sorting first

\[ R_a \xrightarrow{\text{local sort}} R'_a \]
\[ R_b \xrightarrow{\text{local sort}} R'_b \]

\[ R'_1 \]
\[ R'_2 \]
\[ R'_3 \]

result

in order

merge
Parallel/Distributed Join

Input
Relations $R, S$
May or may not be fragmented

Output
$R \bowtie S$
Result at one or more sites
Partitioned Join (Equijoin)

\[ \text{local join} \]

\[ \text{result} \]

\[ f(A) \]

\[ R_A \quad R_B \quad R_1 \quad R_2 \quad R_3 \]

\[ S_1 \quad S_2 \quad S_3 \]

\[ S_A \quad S_B \quad S_C \]
Partitioned Join (Equijoin)

Same partition function $f$ is used for both $R$ and $S$

$f$ can be range or hash partitioning

Local join can be of any type (use any CS 245 optimization)

Various scheduling options, e.g.,

a) Partition $R$; partition $S$; join
b) Partition $R$; build local hash table for $R$; partition $S$ and join
Partitioned Join (Equijoin)

We already know why it works

Sometimes we partition just to make this join possible
Partitioned Join (Equijoin)

Selecting a good partition function $f$ is very important.

Goal is to make all $|R_i| + |S_i|$ the same size.
Asymmetric Fragment + Replicate Join
Asymmetric Fragment + Replicate Join

Can use any partition function $f$ for $R$ (even round robin)

Can do any join, not just equijoin (e.g., $R \bowtie_{R.A < S.B} S$)
General Fragment + Replicate Join

partition
→ 3 fragments

n copies of each fragment
General Fragment + Replicate Join

S is partitioned in similar fashion
General Fragment + Replicate Join

\[ R_1 \times S_1 \]
\[ R_2 \times S_1 \]
\[ R_3 \times S_1 \]
\[ R_1 \times S_2 \]
\[ R_2 \times S_2 \]
\[ R_3 \times S_2 \]

n × m pairings of R, S fragments

result
General Fragment + Replicate Join

The asymmetric F+R join is special case of the general F+R

Asymmetric F+R may be good if S is small

Also works on non-equijoins
Semi-Join

Goal: reduce communication traffic

\[ R_A \bowtie S \Rightarrow (R_A \bowtie S) \bowtie S \text{ or } \]
\[ R_A (S_A \bowtie R) \text{ or } \]
\[ (R_A \bowtie S) \bowtie (S_A \bowtie R) \]
Semi-Join

$R \bowtie S$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
Semi-Join

\[ R \bowtie S \]

\[
\begin{array}{c|c}
A & B \\
2 & a \\
10 & b \\
25 & c \\
30 & d \\
\end{array}
\]

\[
\begin{array}{c|c}
A & C \\
3 & x \\
10 & y \\
15 & z \\
25 & w \\
32 & x \\
\end{array}
\]

\[ \Pi_A R = [2, 10, 25, 30] \]
Semi-Join

\( R \bowtie S \)

\[
\begin{array}{c|c}
A & B \\
\hline
2 & a \\
10 & b \\
25 & c \\
30 & d \\
\end{array}
\hspace{1cm}
\begin{array}{c|c}
A & C \\
\hline
3 & x \\
10 & y \\
15 & z \\
25 & w \\
32 & x \\
\end{array}
\]

\( \prod_A R = [2, 10, 25, 30] \)

\( R \bowtie S \)

\( S \bowtie R = \begin{array}{c|c}
10 & y \\
25 & w \\
\end{array} \)
Semi-Join

Transmitted data
With semi-join $R \bowtie (S \bowtie R)$: $T = 4 |A| + 2 |A + C| + \text{result}$
With join $R \bowtie S$: $T = 4 |A + B| + \text{result}$

better if $|B|$ is large
Semi-Join

Assume $R$ is the smaller relation

Then, in general,

$\left( R \times^A S \right) \bowtie^A S$ is better than $\left( R \times^A S \right)$ if

$\text{size}(\prod_A S) + \text{size}(R \times^A S) < \text{size}(R)$

Can do similar comparison for other joins

→ Only taking into account transmission cost
Semi-Join

Trick: encode $\prod_A S$ (or $\prod_A R$) as a bit vector

keys in $S$ → 001101000010100

← one bit/possible key →

CS 347 Notes 3
Semi-Join

Useful for three way joins $R \bowtie S \bowtie T$
Semi-Join

Useful for three way joins \( R \bowtie S \bowtie T \)

Option 1: \( R' \bowtie S' \bowtie T \) where \( R' = R \bowtie S \)
\[
S' = S \bowtie T
\]
Semi-Join

Useful for three way joins $R \bowtie S \bowtie T$

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

Option 2: $R'' \bowtie S' \bowtie T$ where $R'' = R \bowtie S'$
$S' = S \bowtie T$
Semi-Join

Useful for three way joins $R \bowtie S \bowtie T$

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 ~ exponential in # of relations
Privacy Preserving Join

Site 1 has $R(A, B)$
Site 2 has $S(A, C)$

Want to compute $R \bowtie S$

Site 1 should **not** discover any $S$ info not in the join
Site 2 should **not** discover any $R$ info not in the join
Privacy Preserving Join

Semi-join won’t work
If site 1 sends $\prod_A R$ to site 2, site 2 learns all keys of $R$
Privacy Preserving Join

Fix: send hash keys only

Site 1 hashes each value of $A$ before sending
Site 2 hashes its own $A$ values (same $h$) to see what tuples match

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>a₁</td>
<td>b₁</td>
</tr>
<tr>
<td></td>
<td>a₂</td>
<td>b₂</td>
</tr>
<tr>
<td></td>
<td>a₃</td>
<td>b₃</td>
</tr>
<tr>
<td></td>
<td>a₄</td>
<td>b₄</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>a₁</td>
<td>c₁</td>
</tr>
<tr>
<td></td>
<td>a₃</td>
<td>c₂</td>
</tr>
<tr>
<td></td>
<td>a₅</td>
<td>c₃</td>
</tr>
<tr>
<td></td>
<td>a₇</td>
<td>c₄</td>
</tr>
</tbody>
</table>

\[
\Pi_A R = (h(a₁), h(a₂), h(a₃), h(a₄))
\]

Site 2 sees it has $h(a₁), h(a₃)$
(a₁, c₁), (a₃, c₃)
Privacy Preserving Join

What is the problem?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>a₁</td>
<td>b₁</td>
</tr>
<tr>
<td></td>
<td>a₂</td>
<td>b₂</td>
</tr>
<tr>
<td></td>
<td>a₃</td>
<td>b₃</td>
</tr>
<tr>
<td></td>
<td>a₄</td>
<td>b₄</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>a₁</td>
<td>c₁</td>
</tr>
<tr>
<td></td>
<td>a₃</td>
<td>c₂</td>
</tr>
<tr>
<td></td>
<td>a₅</td>
<td>c₃</td>
</tr>
<tr>
<td></td>
<td>a₇</td>
<td>c₄</td>
</tr>
</tbody>
</table>

\[ \Pi_A R = (h(a_1), h(a_2), h(a_3), h(a_4)) \]

site 2 sees it has \( h(a_1), h(a_3) \)

(a₁, c₁), (a₃, c₃)

site 1

site 2
Privacy Preserving Join

What is the problem?

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a_1 & b_1 \\
a_2 & b_2 \\
a_3 & b_3 \\
a_4 & b_4 \\
\hline
\end{array}
\quad \rightarrow \quad
\begin{array}{|c|c|}
\hline
A & C \\
\hline
a_1 & c_1 \\
a_3 & c_2 \\
a_5 & c_3 \\
a_7 & c_4 \\
\hline
\end{array}
\]

\[\Pi_A R = (h(a_1), h(a_2), h(a_3), h(a_4))\]

site 2 sees it has \(h(a_1), h(a_3)\)

\[\text{site 1} \quad (a_1, c_1), (a_3, c_3)\]

\[\text{site 2}\]

Dictionary attack
Site 2 can take all keys \(a_1, a_2, a_3, ...\) and checks if \(h(a_1), h(a_2), h(a_3), ...\) matches what site 1 sent
Privacy Preserving Join

Our adversary model: honest but curious

Dictionary attack is possible (cheating is internal and can’t be caught)

Sending incorrect keys not possible (cheater could be caught)
Privacy Preserving Join

One solution
*Information Sharing Across Private Databases*. Agrawal et al., SIGMOD 2003

→ Use commutative encryption functions

\[ E_i(x) = x \text{ encrypted using a key private to site } i \]
\[ E_1(E_2(x)) = E_2(E_1(x)) \]

Shorthand: \( E_1(x) \) is \( \bar{x} \), \( E_2(x) \) is \( x \), \( E_1(E_2(x)) \) is \( \bar{x} \)
Privacy Preserving Join

One solution

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>a₁</td>
<td>b₁</td>
</tr>
<tr>
<td></td>
<td>a₂</td>
<td>b₂</td>
</tr>
<tr>
<td></td>
<td>a₃</td>
<td>b₃</td>
</tr>
<tr>
<td></td>
<td>a₄</td>
<td>b₄</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>a₁</td>
<td>c₁</td>
</tr>
<tr>
<td></td>
<td>a₃</td>
<td>c₂</td>
</tr>
<tr>
<td></td>
<td>a₅</td>
<td>c₃</td>
</tr>
<tr>
<td></td>
<td>a₇</td>
<td>c₄</td>
</tr>
</tbody>
</table>

site 1

Computes \((\overline{a₁}, \overline{a₃}, \overline{a₅}, \overline{a₇})\), intersects with \((\overline{a₁}, \overline{a₂}, \overline{a₃}, \overline{a₄})\)

\((a₁, b₁), (a₃, b₃)\)

site 2
Other Privacy Preserving Operations

Inequality join \( R \bowtie_{R.A > S.A} S \)

Similarity join \( R \bowtie_{\text{sim}(R.A,S.A) \leq e} S \)
Other Parallel Operations

Duplicate elimination
Sort first (in parallel) then eliminate duplicates in result
Partition tuples (range or hash) and eliminate locally

Aggregates
Partition by grouping attributes; compute aggregate locally
Parallel/Distributed Aggregates

\begin{align*}
R_a & \quad id \quad dept \quad salary \\
1 & \quad toy \quad 10 \\
2 & \quad toy \quad 20 \\
3 & \quad sales \quad 15 \\
R_b & \quad id \quad dept \quad salary \\
4 & \quad sales \quad 5 \\
5 & \quad toy \quad 20 \\
6 & \quad mgmt \quad 15 \\
7 & \quad sales \quad 10 \\
8 & \quad mgmt \quad 30 \\
\end{align*}

\text{sum (sal) group by dept}
Parallel/Distributed Aggregates

\[
\text{id} \quad \text{dept} \quad \text{salary} \\
1 \quad \text{toy} \quad 10 \\
2 \quad \text{toy} \quad 20 \\
3 \quad \text{sales} \quad 15 \\
\]

\[
\text{id} \quad \text{dept} \quad \text{salary} \\
4 \quad \text{sales} \quad 5 \\
5 \quad \text{toy} \quad 20 \\
6 \quad \text{mgmt} \quad 15 \\
7 \quad \text{sales} \quad 10 \\
8 \quad \text{mgmt} \quad 30 \\
\]

\[
\text{id} \quad \text{dept} \quad \text{salary} \\
1 \quad \text{toy} \quad 10 \\
2 \quad \text{toy} \quad 20 \\
5 \quad \text{toy} \quad 20 \\
6 \quad \text{mgmt} \quad 15 \\
8 \quad \text{mgmt} \quad 30 \\
\]

\[
\text{id} \quad \text{dept} \quad \text{salary} \\
3 \quad \text{sales} \quad 15 \\
4 \quad \text{sales} \quad 5 \\
7 \quad \text{sales} \quad 10 \\
\]

\[\text{sum (sal) group by dept}\]
Parallel/Distributed Aggregates

sum (sal) group by dept

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>toy</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>sales</td>
<td>15</td>
</tr>
</tbody>
</table>

$R_a$

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>sales</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>mgmt</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>sales</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>mgmt</td>
<td>30</td>
</tr>
</tbody>
</table>

$R_b$

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>toy</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>mgmt</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>mgmt</td>
<td>30</td>
</tr>
</tbody>
</table>

$R_a$ + $R_b$

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>sales</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>sales</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>sales</td>
<td>10</td>
</tr>
</tbody>
</table>

sum (sal) group by dept

<table>
<thead>
<tr>
<th>dept</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>toy</td>
<td>50</td>
</tr>
<tr>
<td>mgmt</td>
<td>45</td>
</tr>
<tr>
<td>sales</td>
<td>30</td>
</tr>
</tbody>
</table>
Parallel/Distributed Aggregates

<table>
<thead>
<tr>
<th></th>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>1</td>
<td>toy</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>sales</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rb</td>
<td>4</td>
<td>sales</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>mgmt</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>sales</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>mgmt</td>
<td>30</td>
</tr>
</tbody>
</table>

\[ \text{sum (sal) group by dept with less data} \]
Parallel/Distributed Aggregates

### Table Ra

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>toy</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>sales</td>
<td>15</td>
</tr>
</tbody>
</table>

### Table Rb

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>sales</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>mgmt</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>sales</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>mgmt</td>
<td>30</td>
</tr>
</tbody>
</table>

\[ \text{sum (sal) group by dept with less data} \]
### Parallel/Distributed Aggregates

#### Relation $R_a$

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>toy</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>sales</td>
<td>15</td>
</tr>
</tbody>
</table>

#### Relation $R_b$

<table>
<thead>
<tr>
<th>id</th>
<th>dept</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>sales</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>toy</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>mgmt</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>sales</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>mgmt</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept</th>
<th>salary</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>toy</td>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>toy</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>mgmt</td>
<td>45</td>
<td>45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept</th>
<th>salary</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>sales</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>sales</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

**Sum (sal) group by dept with less data**
Parallel/Distributed Aggregates

Enhancement
Perform aggregate during partition to reduce data transmitted
→ Does not work for all aggregate functions—which ones?
Preview: MapReduce

data A₁

map

data B₁

reduce

data C₁

data A₂

data B₂

data C₁

data A₃
Parallel/Distributed Selection

Straightforward if one can leverage range or hash partitioning

How do indexes work?
Parallel/Distributed Selection

Partition vector can act as the root of a distributed index
Parallel/Distributed Selection

Distributed indexes on a non-partition attributes get complicated
Parallel/Distributed Selection

Indexing schemes
Which one is better in a distributed environment?
How to make updates and expansion efficient?
Where to store the directory and set of participants?
Is global knowledge necessary?

→ If the index is not too big, it may be better to keep it whole and replicate it
Query Processing

Decomposition ✔
Localization ✔

Optimization
Overview ✔
Joins and other operations ✔
Inter-operation parallelism
Optimization strategies
Inter-Operation Parallelism

Pipelined
Independent
Pipelined Parallelism

\[ \sigma_{c} \bowtie R \bowtie S \]

- Site 1: \( \sigma_{c} \)
- Site 2: \( S \)
- Site 1 joins to Site 2
- Result: tuples matching \( \sigma_{c} \)

Diagram:
- Site 1: \( R \)
- Site 2: \( S \)
- Join: \( \bowtie \)
- Probe: \( \leftarrow \)
- Result:
Pipelined Parallelism

Pipelining cannot be used in all cases

\[ \text{stream of R tuples} \qquad \text{stream of S tuples} \]
Independent Parallelism

1. \( \text{temp}_1 \leftarrow R \bowtie S \)
\( \text{temp}_2 \leftarrow T \bowtie V \)
2. \( \text{result} \leftarrow \text{temp}_1 \bowtie \text{temp}_2 \)
Summary

As we consider query plans for optimization, we must consider various *new strategies* for

individual operations

scheduling operations