CS 347: Parallel and Distributed Data Management

Notes 02: Distributed DB Design
Distributed DB Design

Top-down approach:  - have DB...
- how to split and allocate the sites

Multi-DBs (or bottom-up): no design issues!
Two issues in DDB design:

- Fragmentation
- Allocation

Note: issues not independent, but will cover separately
Example

Employee relation E (#, name, loc, sal, ...) 

40% of queries: 
Qa: select * 
   from E 
   where loc=Sa 
   and...

40% of queries: 
Qb: select * 
   from E 
   where loc=Sb 
   and ...
Example

Employee relation $E$ (#, name, loc, sal,...)

40% of queries:  

Qa: select *  
from E  
where loc=Sa  
and...

40% of queries:  

Qb: select *  
from E  
where loc=Sb  
and ...

Motivation: Two sites: Sa, Sb

Qa → Sa  
Sb ← Qb
• It does not take a rocket scientist to figure out fragmentation...
# NM  Loc  Sal
E
5  Joe  Sa  10
7  Sally  Sb  25
8  Tom  Sa  15

#  NM  Loc  Sal
F
7  Sally  Sb  25

At Sa

At Sb
\[ F = \{ F_1, F_2 \} \]

\[ F_1 = \sigma_{\text{loc}=\text{Sa}} E \quad F_2 = \sigma_{\text{loc}=\text{Sb}} E \]
\[ F = \{ F_1, F_2 \} \]

\[ F_1 = \mathcal{O}_{\text{loc}=\text{Sa}} E \quad F_2 = \mathcal{O}_{\text{loc}=\text{Sb}} E \]

\[ \Rightarrow \text{called primary horizontal fragmentation} \]
Fragmentation

- Horizontal
  - Primary
    - depends on local attributes
  - Derived
    - depends on foreign relation

- Vertical
Fragmentation

- Horizontal
  - Primary depends on local attributes
  - Derived depends on foreign relation

- Vertical

Fragmentation also called Sharding
Three common horizontal partitioning techniques

- Round robin
- Hash partitioning
- Range partitioning
• Round robin

<table>
<thead>
<tr>
<th>R</th>
<th>D₀</th>
<th>D₁</th>
<th>D₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>→</td>
<td>t₁</td>
<td></td>
</tr>
<tr>
<td>t₂</td>
<td>→</td>
<td>t₂</td>
<td></td>
</tr>
<tr>
<td>t₃</td>
<td>→</td>
<td>t₃</td>
<td></td>
</tr>
<tr>
<td>t₄</td>
<td>→</td>
<td>t₄</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>→</td>
<td>t₅</td>
<td></td>
</tr>
</tbody>
</table>

• Evenly distributes data
• Good for scanning full relation
• Not good for point or range queries
• Hash partitioning

<table>
<thead>
<tr>
<th></th>
<th>D₀</th>
<th>D₁</th>
<th>D₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>[h(k₁)=2]</td>
<td></td>
<td>[t₁]</td>
</tr>
<tr>
<td>t₂</td>
<td>[h(k₂)=0]</td>
<td>[t₂]</td>
<td></td>
</tr>
<tr>
<td>t₃</td>
<td>[h(k₃)=0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₄</td>
<td>[h(k₄)=1]</td>
<td>[t₄]</td>
<td></td>
</tr>
</tbody>
</table>

... 

• Good for point queries on key; also for joins
• Not good for range queries; point queries not on key
• If hash function good, even distribution
### Range partitioning

<table>
<thead>
<tr>
<th>R</th>
<th>D₀</th>
<th>D₁</th>
<th>D₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1: A=5</td>
<td></td>
<td>→ t1</td>
<td></td>
</tr>
<tr>
<td>t2: A=8</td>
<td>4 7</td>
<td></td>
<td>→ t2</td>
</tr>
<tr>
<td>t3: A=2</td>
<td>V₀ V₁</td>
<td>→ t3</td>
<td></td>
</tr>
<tr>
<td>t4: A=3</td>
<td></td>
<td>→ t4</td>
<td></td>
</tr>
</tbody>
</table>

- Good for some range queries on A
- Need to select good vector: else unbalance
  - → data skew
  - → execution skew
Which are good fragmentations?

Example:

\[
F = \{ F_1, F_2 \}
\]

\[
F_1 = \sigma_{\text{sal}<10} E \quad F_2 = \sigma_{\text{sal}>20} E
\]
Which are good fragmentations?

Example:

\[ F = \{ F_1, F_2 \} \]

\[ F_1 = \sigma_{\text{sal}<10} E \]

\[ F_2 = \sigma_{\text{sal}>20} E \]

⇒ Problem: Some tuples lost!
Which are good fragmentations?

Second example:

\[ F = \{ F_3, F_4 \} \]

\[ F_3 = \sigma_{sal<10} E \] \hspace{1cm} \[ F_4 = \sigma_{sal>5} E \]
Which are good fragmentations?

Second example:

\[ F = \{ F_3, F_4 \} \]

\[ F_3 = \sigma_{\text{sal}<10} E \quad F_4 = \sigma_{\text{sal}>5} E \]

» Tuples with 5 < sal < 10 are duplicated...
⇒ Prefer to deal with replication explicitly

Example: \( F = \{ F_5, F_6, F_7 \} \)

\[
F_5 = \sigma \text{sal \leq 5} \quad E \\
F_6 = \sigma \text{5 < sal < 10} \quad E \\
F_7 = \sigma \text{sal \geq 10} \quad E
\]

† Then replicate \( F_6 \) if convenient
(part of allocation problem)
Desired properties for horizontal fragmentation

\[ R \Rightarrow F = \{ F_1, F_2, \ldots \} \]

(1) Completeness

\[ \forall t \in R, \exists F_i \in F \text{ such that } t \in F_i \]
(2) **Disjointness**

\[ \forall t \in F_i, \neg \exists \ F_j \text{ such that } t \in F_j, \ i \neq j, \ F_i, F_j \in F \]

(3) **Reconstruction** - ignore
How do we get completeness and disjointness?

(1) Check it “manually”!

e.g., \( F_1 = \sigma \text{sal}<10 \ E \); \( F_2 = \sigma \text{sal}\geq10 \ E \)
How do we get completeness and disjointness?

(2) “Automatically” generate fragments with these properties

Desired simple predicates $\Rightarrow$ Fragments
Example of generation

• Say queries use predicates:
  \[ A < 10, \ A > 5, \ \text{Loc} = S_A, \ \text{Loc} = S_B \]

• Next:
  - generate “minterm” predicates
  - eliminate useless ones
Minterm predicates (part I)

(1) \( A<10 \land A>5 \land \text{Loc}=S_A \land \text{Loc}=S_B \)
(2) \( A<10 \land A>5 \land \text{Loc}=S_A \land \neg(\text{Loc}=S_B) \)
(3) \( A<10 \land A>5 \land \neg(\text{Loc}=S_A) \land \text{Loc}=S_B \)
(4) \( A<10 \land A>5 \land \neg(\text{Loc}=S_A) \land \neg(\text{Loc}=S_B) \)
(5) \( A<10 \land \neg(A>5) \land \text{Loc}=S_A \land \text{Loc}=S_B \)
(6) \( A<10 \land \neg(A>5) \land \text{Loc}=S_A \land \neg(\text{Loc}=S_B) \)
(7) \( A<10 \land \neg(A>5) \land \neg(\text{Loc}=S_A) \land \text{Loc}=S_B \)
(8) \( A<10 \land \neg(A>5) \land \neg(\text{Loc}=S_A) \land \neg(\text{Loc}=S_B) \)
Minterm predicates (part I)

(1) \(A < 10 \land A > 5 \land \text{Loc} = S_A \land \text{Loc} = S_B\)

(2) \(A < 10 \land A > 5 \land \text{Loc} = S_A \land \neg(\text{Loc} = S_B)\)

(3) \(A < 10 \land A > 5 \land \neg(\text{Loc} = S_A) \land \text{Loc} = S_B\)

(4) \(A < 10 \land A > 5 \land \neg(\text{Loc} = S_A) \land \neg(\text{Loc} = S_B)\)

(5) \(A < 10 \land \neg(A > 5) \land \text{Loc} = S_A \land \text{Loc} = S_B\)

(6) \(A < 10 \land \neg(A > 5) \land \text{Loc} = S_A \land \neg(\text{Loc} = S_B)\)

(7) \(A < 10 \land \neg(A > 5) \land \neg(\text{Loc} = S_A) \land \text{Loc} = S_B\)

(8) \(A < 10 \land \neg(A > 5) \land \neg(\text{Loc} = S_A) \land \neg(\text{Loc} = S_B)\)
Minterm predicates (part I)

(1) \( A < 10 \land A > 5 \land \text{Loc} = S_A \land \text{Loc} = S_B \)

(2) \( A < 10 \land A > 5 \land \text{Loc} = S_A \land \neg(\text{Loc} = S_B) \)

(3) \( A < 10 \land A > 5 \land \neg(\text{Loc} = S_A) \land \text{Loc} = S_B \)

(4) \( A < 10 \land A > 5 \land \neg(\text{Loc} = S_A) \land \neg(\text{Loc} = S_B) \)

(5) \( A < 10 \land \neg(A > 5) \land \text{Loc} = S_A \land \text{Loc} = S_B \)

(6) \( A < 10 \land \neg(A > 5) \land \text{Loc} = S_A \land \neg(\text{Loc} = S_B) \)

(7) \( A < 10 \land \neg(A > 5) \land \neg(\text{Loc} = S_A) \land \text{Loc} = S_B \)

(8) \( A < 10 \land \neg(A > 5) \land \neg(\text{Loc} = S_A) \land \neg(\text{Loc} = S_B) \)
Minterm predicates (part II)

(9) \neg(A<10) \land A>5 \land \text{Loc}=S_A \land \text{Loc}=S_B
(10) \neg(A<10) \land A>5 \land \text{Loc}=S_A \land \neg((\text{Loc}=S_B)
(11) \neg(A<10) \land A>5 \land \neg((\text{Loc}=S_A) \land \text{Loc}=S_B
(12) \neg(A<10) \land A>5 \land \neg((\text{Loc}=S_A) \land \neg((\text{Loc}=S_B)
(13) \neg(A<10) \land \neg(A>5) \land \text{Loc}=S_A \land \text{Loc}=S_B
(14) \neg(A<10) \land \neg(A>5) \land \text{Loc}=S_A \land \neg((\text{Loc}=S_B)
(15) \neg(A<10) \land \neg(A>5) \land \neg((\text{Loc}=S_A) \land \text{Loc}=S_B
(16) \neg(A<10) \land \neg(A>5) \land \neg((\text{Loc}=S_A) \land \neg((\text{Loc}=S_B)
Minterm predicates (part II)

(9) \( \neg(A<10) \land A>5 \land \text{Loc}=S_A \land \text{Loc}=S_B \)

(10) \( \neg(A<10) \land A>5 \land \text{Loc}=S_A \land \neg(\text{Loc}=S_B) \)

(11) \( \neg(A<10) \land A>5 \land \neg(\text{Loc}=S_A) \land \text{Loc}=S_B \)

(12) \( \neg(A<10) \land A>5 \land \neg(\text{Loc}=S_A) \land \neg(\text{Loc}=S_B) \)

(13) \( \neg(A<10) \land \neg(A>5) \land \text{Loc}=S_A \land \text{Loc}=S_B \)

(14) \( \neg(A<10) \land \neg(A>5) \land \text{Loc}=S_A \land \neg(\text{Loc}=S_B) \)

(15) \( \neg(A<10) \land \neg(A>5) \land \neg(\text{Loc}=S_A) \land \text{Loc}=S_B \)

(16) \( \neg(A<10) \land \neg(A>5) \land \neg(\text{Loc}=S_A) \land \neg(\text{Loc}=S_B) \)

\( A \geq 10 \)
Final fragments:

F2: \( 5 < A < 10 \) \( \land \) Loc=\( S_A \)
F3: \( 5 < A < 10 \) \( \land \) Loc=\( S_B \)
F6: \( A \leq 5 \) \( \land \) Loc=\( S_A \)
F7: \( A \leq 5 \) \( \land \) Loc=\( S_B \)
F10: \( A \geq 10 \) \( \land \) Loc=\( S_A \)
F11: \( A \geq 10 \) \( \land \) Loc=\( S_B \)
Note: elimination of useless fragments depends on application semantics:

e.g.: if LOC could be \( \neq S_A, \neq S_B \),
we need to add fragments

\[
F_4: \quad 5 < A < 10 \quad \land \quad \text{Loc} \neq S_A \land \text{Loc} \neq S_B
\]

\[
F_8: \quad A \leq 5 \quad \land \quad \text{Loc} \neq S_A \land \text{Loc} \neq S_B
\]

\[
F_{12}: \quad A \geq 10 \quad \land \quad \text{Loc} \neq S_A \land \text{Loc} \neq S_B
\]
Why does this work?

Predicates:  

\[ p_1 \land p_2 \land p_3 \land p_4 \]
\[ p_1 \land p_2 \land p_3 \land \neg p_4 \]
\[ \vdash \]
\[ \neg p_1 \land \neg p_2 \land \neg p_3 \land \neg p_4 \]
(1) **Completeness:** Take $t \in R$

$p_i(t)$ must be T or F!

Say $p_1(t) = T$  $p_2(t) = T$  $p_3(t) = F$  $p_4(t) = F$

Then $t$ is in fragment with predicate

$p_1 \land p_2 \land \neg p_3 \land \neg p_4$
(2) Disjointness
Say \( t \in \text{Fragment } p_1 \land p_2 \land \neg p_3 \land \neg p_4 \)
Then:
\( p_1(t) = T, \ p_2(t) = T, \ p_3(t) = F, \ p_4(t) = F \)
\( \Rightarrow t \) cannot be in any other fragment!
Summary

- Given simple predicates \( Pr = \{ p_1, p_2, .. p_m \} \)
  minterm predicates are

\[
M = \{ m \mid m = \bigwedge_{p_k \in Pr} p_k^*, \ 1 \leq k \leq m \}
\]

where \( p_k^* \) is \( p_k \) or is \( \neg p_k \)

- Fragments \( \sigma_m R \) for all \( m \in M \) are complete and disjoint
Another Desired Fragmentation Property:

Match Access Patterns

- Frequently accessed together:
  - data A
  - data B
  - data C

Try to place in same fragment.
Return to example:

E(#, NM, LOC, SAL,...)

Common queries:

Qa: select * from E where LOC=Sa and ...

Qb: select * from E where LOC=Sb and ...
Three choices:

(1) $Pr = \{\} \quad F_1 = \{E\}$

(2) $Pr = \{\text{LOC} = \text{Sa}, \text{LOC} = \text{Sb}\}$

$F_2 = \{\sigma_{\text{loc} = \text{Sa}} E, \sigma_{\text{loc} = \text{Sb}} E\}$

(3) $Pr = \{\text{LOC} = \text{Sa}, \text{LOC} = \text{Sb}, \text{Sal} < 10\}$

$F_3 = \{\sigma_{\text{loc} = \text{Sa} \land \text{sal} < 10} E, \sigma_{\text{loc} = \text{Sa} \land \text{sal} \geq 10} E, \sigma_{\text{loc} = \text{Sb} \land \text{sal} < 10} E, \sigma_{\text{loc} = \text{Sb} \land \text{sal} \geq 10} E\}$
In other words:

\[
\begin{align*}
Q_a & : \text{Select ... \ loc = } S_a \ ... \\
Q_b & : \text{Select ... \ loc = } S_b \ ...
\end{align*}
\]
In other words:

$$\text{Loc}=\text{Sa} \land \text{sal} < 10$$
$$\text{Loc}=\text{Sa} \land \text{sal} \geq 10$$
$$\text{Loc}=\text{Sb} \land \text{sal} < 10$$
$$\text{Loc}=\text{Sb} \land \text{sal} \geq 10$$

$F_1$

$F_2$

$F_3$

$Q_a$: Select ... loc = Sa ...
$Q_b$: Select ... loc = Sb ...

$F_2$ is good...
(not $F_1$, $F_3$)
Derived horizontal fragmentation

Example:

\[ E(#, \text{NM, SAL, LOC}) \]

\[ F = \{ E_1, E_2 \} \text{ by LOC} \]

\[ J(#, \text{DES, ...}) \]

Common query for project:

[Given employee name,
list projects (s)he works in]
<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Joe</td>
<td>Sa</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Tom</td>
<td>Sa</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(at S_a)

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Fred</td>
<td>Sb</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(at S_b)

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>work on 347 hw</td>
</tr>
<tr>
<td>7</td>
<td>go to moon</td>
</tr>
<tr>
<td>5</td>
<td>build table</td>
</tr>
<tr>
<td>12</td>
<td>rest</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>NM</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>5</td>
<td>Joe</td>
</tr>
<tr>
<td>8</td>
<td>Tom</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

*(at Sa)*

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Fred</td>
<td>Sb</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*(at Sb)*

<table>
<thead>
<tr>
<th>#</th>
<th>Des</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>work on 347 hw</td>
</tr>
<tr>
<td>5</td>
<td>build table</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

\[ J_1 = J \times E_1 \]

<table>
<thead>
<tr>
<th>#</th>
<th>Des</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>go to moon</td>
</tr>
<tr>
<td>12</td>
<td>rest</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

\[ J_2 = J \times E_2 \]
Derived horizontal fragmentation

\[ R, F = \{ F_1, F_2, \ldots, F_n \} \]

\[ \Downarrow \]

\[ S, D = \{ D_1, D_2, \ldots, D_n \} \text{ where } D_i = S \bowtie F_i \]

**Convention:** R is owner

S is member

F could be primary or derived
• Checking completeness and disjointness of derived fragmentation

Example: Say J is:

<table>
<thead>
<tr>
<th>#</th>
<th>Des</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>build chair</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

* But no # = 33 in E₁ nor in E₂!

This J tuple will not be in J₁ nor J₂
Fragmentation not complete
Need to enforce referential integrity constraint:

join attr(#) of member relation

↓

joint attr(#) of owner relation
Example:

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Joe</td>
<td>Sa</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Fred</td>
<td>Sb</td>
<td>20</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>day off</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**J**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>day off</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**J1**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>day off</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**J2**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>day off</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Fragmentation is not disjoint!**
To get disjointness

Join attribute(#) should be key of owner relation
Summary: horizontal fragmentation

- Type: primary, derived
- Properties: completeness, disjointness
**Vertical fragmentation**

Example:

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Joe</td>
<td>Sa</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Fred</td>
<td>Sa</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
E1

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Joe</td>
<td>Sa</td>
</tr>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
</tr>
<tr>
<td>8</td>
<td>Fred</td>
<td>Sa</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E2

<table>
<thead>
<tr>
<th>#</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
\[ R[T] \implies R_1[T_1] \quad T_i \subseteq T \]
\[ \vdots \]
\[ R_n[T_n] \]

\[ \rightarrow \] Just like normalization of relations
Properties: \( R[T] \implies R_i[T_i] \)

(1) Completeness

\[ \bigcup_i T_i = T \]

all \( i \)
(2) Disjointness

\[ T_i \cap T_j = \emptyset \quad \text{for all } i, j \quad i \neq j \]

\[ E(\#, \text{LOC}, \text{SAL}) \rightarrow E_1(\#, \text{LOC}) \rightarrow E_2(\text{SAL}) \]
(2) Disjointness

\[ T_i \cap T_j = \emptyset \text{ for all } i, j \neq j \]

\[ E(#, LOC, SAL) \]

Not a desirable property!!
(could not reconstruct R!!)
(3) Lossless join

\[ R_i = R \]

all \( i \)

One way to achieve lossless join:
Repeat key in all fragments, i.e.,
\[ \text{Key} \subseteq T_i \text{ for all } i \]
How do we decide what attributes are grouped with which?

Example:

E(#,NM,LOC,SAL)

\[ E_1(#,NM,LOC) \]
\[ E_2(#,SAL) \]

?
### Attribute affinity matrix

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A2</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A3</td>
<td>45</td>
<td>48</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>75</td>
<td>-</td>
</tr>
</tbody>
</table>
Attribute affinity matrix

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A2</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A3</td>
<td>45</td>
<td>48</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>75</td>
<td>-</td>
</tr>
</tbody>
</table>

\[ R_1[K,A_1,A_2,A_3] \quad R_2[K,A_4,A_5] \]
• Textbook (Ozsu & Valduriez) discusses
  – How to build affinity matrix
  – How to identify attribute clusters
  – How to partition relation
Allocation

Example: \( E(\#, NM, LOC, SAL) \Rightarrow \)

\[ F_1 = \sigma_{loc=Sa} E ; F_2 = \sigma_{loc=Sb} E \]

Qa: select ... where loc=Sa...
Qb: select ... where loc=Sb...

Where do \( F_1, F_2 \) go?

Site a

Site b
Issues

• Where do queries originate
• What is communication cost? and size of answers, relations,...
• What is storage capacity, cost at sites? and size of fragments?
• What is processing power at sites?
More Issues

• What is query processing strategy?
  – How are joins done?
  – Where are answers collected?
Do we replicate fragments?

- Cost of updating copies?
- Writes and concurrency control?
- ...

Optimization problem:

• What is best placement of fragments and/or best number of copies to:
  – minimize query response time
  – maximize throughput
  – minimize “some cost”
  – ...

• Subject to constraints?
  – Available storage
  – Available bandwidth, power,...
  – Keep 90% of response time below X
  – ...

Optimization problem:

• What is best placement of fragments and/or best number of copies to:
  – minimize query response time
  – maximize throughput
  – minimize “some cost”
  – ...

• Subject to constraints?
  – Available storage
  – Available bandwidth, power,...
  – Keep 90% of response time below X
  – ...

This is an incredibly hard problem
Example: Single fragment F

Read cost: \[ \sum_{i=1}^{m} [t_i \times \text{MIN } C_{ij}] \]

i: Originating site of request

ti: Read traffic at Si

Cij: Retrieval cost

Accessing fragment F at Sj from Si
Scenario - Read cost

Stream of read requests for $F_t$ REQ/SEC

$C = \infty$
Write cost

\[ \sum_{i=1}^{m} \sum_{j=1}^{m} X_j \ u_i \ C'_{ij} \]

i: Originating site of request
j: Site being updated
X_j: \begin{cases} 0 & \text{if F not stored at } S_j \\ 1 & \text{if F stored at } S_j \end{cases}
u_i: Write traffic at S_i
C’_{ij}: Write cost

Updating F at S_j from S_i
Scenario - write cost

Updates
ui updates/sec
Storage cost:

\[ \sum_{i=1}^{m} X_i d_i \]

\(X_i:\) \begin{cases} 0 & \text{if F not stored at } S_i \\ 1 & \text{if F stored at } S_i \end{cases}

\(d_i:\) storage cost at \(S_i\)
Target function:

\[
\min \left\{ \sum_{i=1}^{m} [t_i \times \text{MIN } C_{ij} + \sum_{j=1}^{m} X_j \times u_i \times C'_{ij}] + \sum_{i=1}^{m} X_i \times d_i \right\}
\]
Can add more complications:

Examples:
- Multiple fragments
- Fragment sizes
- Concurrency control cost
Case Study: PNUTS

• Where in the World is My Data?
  Sudarshan Kadambi, Jianjun Chen, Brian F. Cooper, David Lomax, Raghu Ramakrishnan, Adam Silberstein, Erwin Tam, Hector Garcia-Molina; VLDB 2011

• Distributed object/tuple store for Yahoo!
Case Study: PNUTS

- Issue: Where to locate data
- Issue: What and where to replicate
PNUTS Discussion

• Dynamic vs Static fragment placement
• Caching vs Replication
Policy Constraints

- **MIN_COPIES**: The minimum number of full replicas of the record that must exist.
- **INCL_LIST**: An inclusion list -- the locations where a full replica of the record must exist.
- **EXCL_LIST**: An exclusion list -- the locations where a full replica of the record cannot exist.
Example Rule

• Rule 1:
• IF TABLE_NAME = "Users"
  THEN
    SET 'MIN_COPIES' = 2
    CONSTRAINT_PRI = 0
Another Example Rule

- Rule 2:
  - IF TABLE_NAME = "Users" AND FIELD STR('home location') = 'France'
    THEN
      SET 'MIN_COPIES' = 3 AND
      SET 'EXCL LIST' = 'USWest, USEast'
      CONSTRAINT PRI = 1
Summary

- Description of fragmentation
- Good fragmentations
- Design of fragmentation
- Allocation