Logistics

Gradiance
No action items for now
Detailed instructions coming shortly
First quiz to be released next Monday (April 4)

Exams
Open book, open laptop
Midterm in class on Wednesday, April 27
Final starting 12:15pm on Friday, June 3
Logistics

Polls
Lecture pace: just right (86%)
Printed handouts: mostly no (77%) → will keep bringing some
Review session: yes (52%), focus more on implementation (60%)
CA office hours
Distributed Database Design

**Bottom-up approach**
Multi-databases
Individual
No design issues!

**Top-down approach**
Start with a blank slate
Similar to centralized DB design
New distribution issues
New Issues

**Fragmentation**
How to split the data?

**Allocation**
Where should each fragment go?

→ Not independent, but will cover them separately
Example

Employee relation
E(id, name, salary, location, ...)

40% of queries
QA: select *
   from E
   where location = A and ...

40% of queries
QB: select *
   from E
   where location = B and ...

Example

Employee relation
E(id, name, salary, location, ...)

40% of queries
Q_A: select *
    from E
    where location = A and ...

40% of queries
Q_B: select *
    from E
    where location = B and ...

Motivation: two sites

Q_A → A → B ← Q_B
Fragmentation

Straightforward...
Fragmentation

Straightforward...

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Ben</td>
<td>A</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Ben</td>
<td>A</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
</tbody>
</table>
Fragmentation

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

\[ F_1 = \sigma_{\text{location}=A} E \]

\[ F_2 = \sigma_{\text{location}=B} E \]
Fragmentation

$F = \{ F_1, F_2 \}$

$F_1 = \sigma_{\text{location}=A}E$

$F_2 = \sigma_{\text{location}=B}E$

→ Primary horizontal fragmentation
Fragmentation

Horizontal

Primary
Using predicates on $E$ only

Derived
Using foreign relation predicates

Vertical
Horizontal Partitioning Techniques

Round robin partitioning

Hash partitioning

Range partitioning
Round Robin Partitioning

Distributes data evenly
Good for scanning full relation
Not good for point or range queries
Hash Partitioning

Distributes data evenly if hash function is good
Good for point queries on key and joins
Not good for range queries and point queries not on key
Range Partitioning

Good for some range queries on partition attribute
Need to select good vector to avoid data/execution skew
Good Fragmentations?

Example 1:

\( F = \{ F_1, F_2 \} \)

\( F_1 = \sigma_{\text{salary}<10}\text{E} \)

\( F_2 = \sigma_{\text{salary}>20}\text{E} \)
Good Fragmentations?

Example 1:

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

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

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

→ Problem: some tuples are lost \((10 \leq \text{salary} < 20)\)
Good Fragmentations?

Example 2:

\( F = \{ F_3, F_4 \} \)

\( F_3 = \sigma_{\text{salary}<10} E \)

\( F_4 = \sigma_{\text{salary}>5} E \)
Good Fragmentations?

Example 2:

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

\[ F_3 = \sigma_{\text{salary} < 10} E \]

\[ F_4 = \sigma_{\text{salary} > 5} E \]

→ Problem: some tuples are duplicated \((5 < \text{salary} < 10)\)
Good Fragmentations?

Prefer to deal with replication separately:

First fragment...

\[ F = \{ F_5, F_6, F_7 \} \]

\[ F_5 = \sigma_{\text{salary} \leq 5} E \]

\[ F_6 = \sigma_{5 < \text{salary} < 10} E \]

\[ F_7 = \sigma_{\text{salary} \geq 10} E \]

... then replicate, e.g., \( F_6 \) if necessary (as part of allocation)
Fragmentation Properties

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

Desired properties of horizontal fragmentations:

1. Completeness
   \[ \forall t \in R, \exists F_i \in \mathbf{F}: t \in F_i \]

2. Disjointness
   \[ \forall t \in F_i, \neg \exists F_j: t \in F_j, i \neq j, \{ F_i, F_j \} \subset \mathbf{F} \]

3. Reconstruction
   \[ R = \bigcup_{F_i \in \mathbf{F}} F_i \]
Fragmentation Properties

How to ensure completeness and disjointness?

(1) Check manually

\[ F_1 = \sigma_{\text{salary} \leq 5} E \quad F_2 = \sigma_{\text{salary} > 5} E \]
Fragmentation Properties

How to ensure completeness and disjointness?

(1) Check manually

\[ F_1 = \sigma_{\text{salary} \leq 5} E \quad F_2 = \sigma_{\text{salary} > 5} E \quad \checkmark \]

(2) Automatically generate fragments with such properties
Minterm Predicates

(1) Consider simple predicates used in queries

E.g., salary < 10, salary > 5, location = A, location = B

(2) Generate minterm predicates

(3) Eliminate useless ones
Minterm Predicates

(1) \( \text{sal} < 10 \land \text{sal} > 5 \land \text{loc} = A \land \text{loc} = B \)
(2) \( \text{sal} < 10 \land \text{sal} > 5 \land \text{loc} = A \land \neg (\text{loc} = B) \)
(3) \( \text{sal} < 10 \land \text{sal} > 5 \land \neg (\text{loc} = A) \land \text{loc} = B \)
(4) \( \text{sal} < 10 \land \text{sal} > 5 \land \neg (\text{loc} = A) \land \neg (\text{loc} = B) \)
(5) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \text{loc} = A \land \text{loc} = B \)
(6) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \text{loc} = A \land \neg (\text{loc} = B) \)
(7) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \neg (\text{loc} = A) \land \text{loc} = B \)
(8) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \neg (\text{loc} = A) \land \neg (\text{loc} = B) \)
Minterm Predicates

(1) \( \text{sal} < 10 \land \text{sal} > 5 \land \text{loc} = \text{A} \land \text{loc} = \text{B} \)

(2) \( \text{sal} < 10 \land \text{sal} > 5 \land \text{loc} = \text{A} \land \neg(\text{loc} = \text{B}) \)

(3) \( \text{sal} < 10 \land \text{sal} > 5 \land \neg(\text{loc} = \text{A}) \land \text{loc} = \text{B} \)

(4) \( \text{sal} < 10 \land \text{sal} > 5 \land \neg(\text{loc} = \text{A}) \land \neg(\text{loc} = \text{B}) \)

(5) \( \text{sal} < 10 \land \neg(\text{sal} > 5) \land \text{loc} = \text{A} \land \text{loc} = \text{B} \)

(6) \( \text{sal} < 10 \land \neg(\text{sal} > 5) \land \text{loc} = \text{A} \land \neg(\text{loc} = \text{B}) \)

(7) \( \text{sal} < 10 \land \neg(\text{sal} > 5) \land \neg(\text{loc} = \text{A}) \land \text{loc} = \text{B} \)

(8) \( \text{sal} < 10 \land \neg(\text{sal} > 5) \land \neg(\text{loc} = \text{A}) \land \neg(\text{loc} = \text{B}) \)
Minterm Predicates

(1) \( \text{sal} < 10 \land \text{sal} > 5 \land \text{loc} = A \land \text{loc} = B \)

(2) \( \text{sal} < 10 \land \text{sal} > 5 \land \text{loc} = A \land \neg (\text{loc} = B) \)

(3) \( \text{sal} < 10 \land \text{sal} > 5 \land \neg (\text{loc} = A) \land \text{loc} = B \)

(4) \( \text{sal} < 10 \land \text{sal} > 5 \land \neg (\text{loc} = A) \land \neg (\text{loc} = B) \)

(5) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \text{loc} = A \land \text{loc} = B \)

(6) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \text{loc} = A \land \neg (\text{loc} = B) \)

(7) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \neg (\text{loc} = A) \land \text{loc} = B \)

(8) \( \text{sal} < 10 \land \neg (\text{sal} > 5) \land \neg (\text{loc} = A) \land \neg (\text{loc} = B) \)
Minterm Predicates

(9) \( \neg (sal < 10) \land sal > 5 \land loc = A \land loc = B \)

(10) \( \neg (sal < 10) \land sal > 5 \land loc = A \land \neg (loc = B) \)

(11) \( \neg (sal < 10) \land sal > 5 \land \neg (loc = A) \land loc = B \)

(12) \( \neg (sal < 10) \land sal > 5 \land \neg (loc = A) \land \neg (loc = B) \)

(13) \( \neg (sal < 10) \land \neg (sal > 5) \land loc = A \land loc = B \)

(14) \( \neg (sal < 10) \land \neg (sal > 5) \land loc = A \land \neg (loc = B) \)

(15) \( \neg (sal < 10) \land \neg (sal > 5) \land \neg (loc = A) \land loc = B \)

(16) \( \neg (sal < 10) \land \neg (sal > 5) \land \neg (loc = A) \land \neg (loc = B) \)
Minterm Predicates

(9) \neg(sal<10) \land sal>5 \land loc=A \land loc=B

(10) \neg(sal<10) \land sal>5 \land loc=A \land \neg(loc=B)

(11) \neg(sal<10) \land sal>5 \land \neg(loc=A) \land loc=B

(12) \neg(sal<10) \land sal>5 \land \neg(loc=A) \land \neg(loc=B)

(13) \neg(sal<10) \land \neg(sal>5) \land loc=A \land loc=B

(14) \neg(sal<10) \land \neg(sal>5) \land loc=A \land \neg(loc=B)

(15) \neg(sal<10) \land \neg(sal>5) \land \neg(loc=A) \land loc=B

(16) \neg(sal<10) \land \neg(sal>5) \land \neg(loc=A) \land \neg(loc=B)
Minterm Predicates

\[ \neg (\text{sal} < 10) \land \text{sal} > 5 \land \text{loc} = A \land \text{loc} = B \]

\[ \neg (\text{sal} < 10) \land \text{sal} > 5 \land \text{loc} = A \land \neg (\text{loc} = B) \]

\[ \neg (\text{sal} < 10) \land \text{sal} > 5 \land \neg (\text{loc} = A) \land \text{loc} = B \]

\[ \neg (\text{sal} < 10) \land \neg (\text{sal} > 5) \land \text{loc} = A \land \text{loc} = B \]

\[ \neg (\text{sal} < 10) \land \neg (\text{sal} > 5) \land \neg (\text{loc} = A) \land \text{loc} = B \]

\[ \neg (\text{sal} < 10) \land \neg (\text{sal} > 5) \land \neg (\text{loc} = A) \land \neg (\text{loc} = B) \]
Minterm Predicates

Final fragments:

\[ F_2 \quad 5 < \text{salary} < 10 \land \text{location} = A \]
\[ F_3 \quad 5 < \text{salary} < 10 \land \text{location} = B \]
\[ F_6 \quad \text{salary} \leq 5 \land \text{location} = A \]
\[ F_7 \quad \text{salary} \leq 5 \land \text{location} = B \]
\[ F_{10} \quad \text{salary} \geq 10 \land \text{location} = A \]
\[ F_{11} \quad \text{salary} \geq 10 \land \text{location} = B \]
Minterm Predicates

Elimination of useless fragments depends on application semantics

E.g., if location could be other than A or B, must add fragments:

\[ F_4 \quad 5 < \text{salary} < 10 \land \text{location} \neq A \land \text{location} \neq B \]
\[ F_8 \quad \text{salary} \leq 5 \land \text{location} \neq A \land \text{location} \neq B \]
\[ F_{12} \quad \text{salary} \geq 10 \land \text{location} \neq A \land \text{location} \neq B \]
Minterm Predicates

Why does this technique work?

To illustrate, consider \( p_1, p_2, p_3, p_4 \)

Predicates

\[
\begin{align*}
p_1 \land p_2 \land p_3 \land p_4 \\
p_1 \land p_2 \land p_3 \land \neg p_4 \\
\vdots & \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\neg p_1 \land \neg p_2 \land \neg p_3 \land \neg p_4
\end{align*}
\]
Minterm Predicates

(1) Completeness

∀ t ∈ R: p_i(t) is either true or false

Say, p_1(t) and p_2(t) are true; p_3(t) and p_4(t) are false

Then t is in the fragment with predicate p_1 ∧ p_2 ∧ ¬p_3 ∧ ¬p_4
Minterm Predicates

(2) Disjointness

Say $t$ is in the fragment with predicate $p_1 \land p_2 \land \neg p_3 \land \neg p_4$

Then $p_1(t)$ and $p_2(t)$ are true; $p_3(t)$ and $p_4(t)$ are false

There is no other fragment $t$ could be in
Minterm Predicates

Summary
Given a set of simple predicates $P = \{ p_1, p_2, ..., p_n \}$, minterm predicates are

$$M = \{ m \mid m = \bigwedge_{1 \leq k \leq n} p_k^* \}$$

where $p_k^*$ is either $p_k$ or $\neg p_k$

$\rightarrow$ Fragments $\sigma_m R$ for all $m \in M$ are complete and disjoint
Matching Access Patterns

Another desired fragmentation property

frequently accessed together

Data A
Data B
Data C

try to place in same fragment
Matching Access Patterns

Earlier example:

\[ E(id, \text{name}, \text{salary}, \text{location}, \ldots) \]

40% of queries
\[ Q_A: \text{select} * \]
\[ \text{from } E \]
\[ \text{where location} = A \text{ and } \ldots \]

40% of queries
\[ Q_B: \text{select} * \]
\[ \text{from } E \]
\[ \text{where location} = B \text{ and } \ldots \]
Matching Access Patterns

Consider the following options:

(1) \( P = \{ \} \)
    \[ F_1 = \{ E \} \]

(2) \( P = \{ \text{location} = A, \text{location} = B \} \)
    \[ F_2 = \{ \sigma_{\text{location}=A} E, \sigma_{\text{location}=B} E \} \]

(3) \( P = \{ \text{location} = A, \text{location} = B, \text{salary} < 10 \} \)
    \[ F_3 = \{ \sigma_{\text{location}=A \land \text{salary}<10} E, \sigma_{\text{location}=A \land \text{salary}\geq10} E, \]
    \[ \sigma_{\text{location}=B \land \text{salary}<10} E, \sigma_{\text{location}=B \land \text{salary}\geq10} E \} \]
Matching Access Patterns

In other words:

\[
\begin{align*}
F_1 & \quad \text{location} = A \land \text{salary} < 10 \\
& \quad \text{location} = A \land \text{salary} \geq 10 \\
& \quad \text{location} = B \land \text{salary} < 10 \\
& \quad \text{location} = B \land \text{salary} \geq 10 \\
\end{align*}
\]
Matching Access Patterns

In other words:

\[
\begin{align*}
F_1 & \quad \text{location} = A \land \text{salary} < 10 \\
F_2 & \quad \text{location} = A \land \text{salary} \geq 10 \\
F_3 & \quad \text{location} = B \land \text{salary} < 10 \\
F_4 & \quad \text{location} = B \land \text{salary} \geq 10
\end{align*}
\]

\[Q_A = \text{select ... location}=A ...\]

\[Q_B = \text{select ... location}=B ...\]
Matching Access Patterns

In other words:

\[ F_1 = \begin{cases} 
\text{location} = A \land \text{salary} < 10 \\
\text{location} = A \land \text{salary} \geq 10 \\
\text{location} = B \land \text{salary} < 10 \\
\text{location} = B \land \text{salary} \geq 10 
\end{cases} \]

\[ F_2 \rightarrow F_3 \]

\[ Q_A = \text{select ... location=A ...} \]

\[ Q_B = \text{select ... location=B ...} \]

\[ F_2 \text{ is good (not } F_1 \text{ and } F_3) \]
Derived Horizontal Fragmentation

Example:

\[ E(\text{id}, \text{name}, \text{salary}, \text{location}, ... ) \]
\[ F = \{ F_1, F_2 \} \text{ by location} \]

\[ T(\text{id}, \text{task}, ...) \]

**Common query for task:**
Given employee name, list tasks (s)he works on
# Derived Horizontal Fragmentation

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Ben</td>
<td>A</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>Max</td>
<td>B</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>design</td>
</tr>
<tr>
<td>1</td>
<td>build</td>
</tr>
<tr>
<td>2</td>
<td>advertise</td>
</tr>
<tr>
<td>4</td>
<td>sell</td>
</tr>
</tbody>
</table>
Derived Horizontal Fragmentation

\[ T_1 = T \times E_1 \]

\[ T_2 = T \times E_2 \]
Derived Horizontal Fragmentation

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

\[ \Downarrow \]

\[ S, \mathbf{G} = \{ G_1, G_2, \ldots \}, G_i = S \times F_i \]

**Convention**

- \( R \) is owner
- \( S \) is member

\( F \) could be primary or derived
Derived Horizontal Fragmentation

Checking completeness and disjointness

Example:

<table>
<thead>
<tr>
<th>id</th>
<th>task</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>test</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

but no id = 6 in $E_1$ or in $E_2$

this $T$ tuple will not be in either $T_1$ or $T_2$

fragmentation is not complete
Derived Horizontal Fragmentation

To get completeness need to enforce referential integrity:

Join attribute of the member relation

  Each value must be present

Join attribute of the owner relation
Derived Horizontal Fragmentation

Example:

\[ E_1 \]

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ E_2 \]

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ T \]

<table>
<thead>
<tr>
<th>id</th>
<th>task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

\[ T_1 \]

<table>
<thead>
<tr>
<th>id</th>
<th>task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

\[ T_2 \]

<table>
<thead>
<tr>
<th>id</th>
<th>task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{fragmentation} \not\text{ disjoint} \]
Derived Horizontal Fragmentation

To get disjointness:

Join attribute should be key of owner relation
Horizontal Fragmentation Summary

Types
Primary
Derived

Properties
Completeness
Disjointness
Vertical Fragmentation

Example:

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Ben</td>
<td>A</td>
<td>21</td>
</tr>
</tbody>
</table>

\[ E \]

\[ E_1 \]

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>Ann</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>Ben</td>
<td>A</td>
</tr>
</tbody>
</table>

\[ E_2 \]

<table>
<thead>
<tr>
<th>id</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
</tr>
</tbody>
</table>
Vertical Fragmentation

Just like normalization of relations

\[ R [ A ] \Rightarrow R_1 [ A_1 ] \]
\[ \vdots \]
\[ R_n [ A_n ] \]

\[ A = \{ a_1, a_2, ..., a_n \} \] set of attributes

\[ A_i \subseteq A \]
Vertical Fragmentation

Properties

\[ R[A] \Rightarrow R_i[A_i] \]

(1) Completeness

\[ \bigcup A_i = A \quad \forall i \]
Vertical Fragmentation

(2) Disjointness

\[ A_i \cap A_j = \emptyset \quad \forall i, j: i \neq j \]
Vertical Fragmentation

(2) Disjointness

$$A_i \cap A_j = \emptyset \quad \forall i, j : i \neq j$$

Not a desirable property (could not reconstruct R)
Vertical Fragmentation

(3) Lossless join

\[ R_i = R \quad \forall i \]

One way to achieve lossless join
Repeat key in all fragments

\[ \text{key} \subseteq A_i \quad \forall i \]
Vertical Fragmentation

How do we decide what attributes are grouped with which?

Example:

\[ E = (id, name, location, salary) \]
### Attribute Affinity Matrix

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$a_2$</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$a_3$</td>
<td>45</td>
<td>48</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$a_4$</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$a_5$</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>

\[
R_1 [ k, a_1, a_2, a_3 ] \quad R_2 [ k, a_4, a_5 ]
\]
Distributed DB Design Issues

Fragmentation ✔
How to split the data?

Allocation
Where should each fragment go?
Allocation

Example:
E (id, name, location, salary), \( F_1 = \sigma_{\text{location}=A}E \), \( F_2 = \sigma_{\text{location}=B}E \)

\[
\begin{align*}
Q_A: & \quad \text{select } * \\
& \quad \text{from } E \\
& \quad \text{where location = A and } \ldots
\end{align*}
\]

\[
\begin{align*}
Q_B: & \quad \text{select } * \\
& \quad \text{from } E \\
& \quad \text{where location = B and } \ldots
\end{align*}
\]

Where do \( F_1, F_2 \) go?

Site A

Site B
Issues

Where do queries originate?
What is the communication cost?
  Size of answers? Relations?
What is the storage capacity and cost at sites?
  Size of fragments?
What is the processing power at sites?
What is the query processing strategy?
  How are joins done?
Where are answers collected?
Issues

Do we replicate *fragments*?
  Cost of updating copies?
  How to provide concurrency control?
Optimization

Best placement of fragments and/or best number of copies to

Minimize query response time
Maximize throughput
Minimize some other cost (e.g., communication)

Subject to constraints

Available storage
Available bandwidth, power, etc.
Other, e.g., keep 90\textsuperscript{th} percentile response time $< x$
Optimization

Best placement of fragments and/or best number of copies to
- Minimize query response time
- Maximize throughput
- Minimize some other cost (e.g., communication)

Subject to constraints
- Available storage
- Available bandwidth, power, etc.
- Other, e.g., keep 90th percentile response time < x

This is an incredibly hard problem
Optimization

Example

Single fragment $F$

Read cost $\sum_{i=1}^{m} \left[ t_i \times \min_j C_{ij} \right]$

- $i$: originating site of request
- $t_i$: read traffic at $S_i$
- $C_{ij}$: retrieval cost (accessing fragment $F$ at $S_j$ from $S_i$)
Optimization

Read cost scenario

\[ C = \infty \]

\[ C_{i,1} \]

\[ C_{i,2} \]

\[ C = \infty \]

\[ C = \infty \]

\[ t_i \]
Optimization

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

- \( i \): originating site of request
- \( j \): site being updated
- \( x_{ij} \): 0 if \( F \) not stored at \( S_j \)
  
  1 if \( F \) stored at \( S_j \)
- \( u_i \): write traffic at \( S_i \)
- \( C'_{ij} \): write cost (updating \( F \) at \( S_j \) from \( S_i \))
Optimization

Write cost scenario

\[ F \xrightarrow{} i \xleftarrow{} F \xrightarrow{} F \xrightarrow{} u_i \]
Optimization

Storage cost  \[ \sum_{i=1}^{m} x_i D_i \]

\[ x_i \]  0 if F not stored at \( S_i \)
1 if F stored at \( S_i \)

\[ D_i \]  storage cost at \( S_i \)
Optimization

Target function

\[
\min \left\{ \sum_{i=1}^{m} \left[ t_i \times \min_j C_{ij} + \sum_{j=1}^{m} x_j \times u_i \times C'_{ij} \right] + \sum_{i=1}^{m} x_i \times D_i \right\}
\]
Optimization

Can add more complications
Multiple fragments
Fragment sizes
Concurrency control cost
Case Study: PNUTS

Distributed object/tuple store for Yahoo!

*Where in the World is My Data?* Kadambi et al., VLDB 2011
Case Study: PNUTS

Issues
Where to locate data
What and where to replicate
Case Study: PNUTS

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
Case Study: PNUTS

Example rule 1

IF

    TABLE_NAME = 'Users'

THEN

    SET 'MIN_COPIES' = 2
    CONSTRAINT_PRI = 0
Case Study: PNUTS

Example 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