Design Problem

■ In the general setting:

Making decisions about the placement of data and programs across the sites of a computer network as well as possibly designing the network itself.

■ In Distributed DBMS, the placement of applications entails

→ placement of the distributed DBMS software; and
→ placement of the applications that run on the database
Dimensions of the Problem

- Access pattern behavior
  - dynamic
  - static
- Level of sharing
- Level of knowledge
  - partial information
  - complete information
- Data
- Data + program

Distribution Design

- **Top-down**
  - mostly in designing systems from scratch
  - mostly in homogeneous systems
- **Bottom-up**
  - when the databases already exist at a number of sites
Top-Down Design

Distribution Design Issues

- Why fragment at all?
- How to fragment?
- How much to fragment?
- How to test correctness?
- How to allocate?
- Information requirements?
Fragmentation

- Can’t we just distribute relations?
- What is a reasonable unit of distribution?
  - relation
    - views are subsets of relations ⇝ locality
    - extra communication
  - fragments of relations (sub-relations)
    - concurrent execution of a number of transactions that access different portions of a relation
    - views that cannot be defined on a single fragment will require extra processing
    - semantic data control (especially integrity enforcement) more difficult

Fragmentation Alternatives – Horizontal

PROJ₀¹ : projects with budgets less than $200,000
PROJ₀² : projects with budgets greater than or equal to $200,000

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>150000</td>
<td>Montreal</td>
</tr>
<tr>
<td>P2</td>
<td>Database</td>
<td>135000</td>
<td>New York</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>New York</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>310000</td>
<td>Paris</td>
</tr>
<tr>
<td>P5</td>
<td>CAD/CAM</td>
<td>500000</td>
<td>Boston</td>
</tr>
</tbody>
</table>
Fragmentation Alternatives – Vertical

PROJ₁: information about project budgets

<table>
<thead>
<tr>
<th>PNO</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>150000</td>
<td>Montreal</td>
</tr>
<tr>
<td>P2</td>
<td>135000</td>
<td>New York</td>
</tr>
<tr>
<td>P3</td>
<td>250000</td>
<td>New York</td>
</tr>
<tr>
<td>P4</td>
<td>310000</td>
<td>Paris</td>
</tr>
<tr>
<td>P5</td>
<td>500000</td>
<td>Boston</td>
</tr>
</tbody>
</table>

PROJ₂: information about project names and locations

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>Montreal</td>
</tr>
<tr>
<td>P2</td>
<td>Database Develop.</td>
<td>New York</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>New York</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>Paris</td>
</tr>
<tr>
<td>P5</td>
<td>CAD/CAM</td>
<td>Boston</td>
</tr>
</tbody>
</table>

Degree of Fragmentation

finite number of alternatives

<table>
<thead>
<tr>
<th>tuples or attributes</th>
</tr>
</thead>
</table>

Finding the suitable level of partitioning within this range
Correctness of Fragmentation

- **Completeness**
  - Decomposition of relation $R$ into fragments $R_1, R_2, ..., R_n$ is complete if and only if each data item in $R$ can also be found in some $R_i$.

- **Reconstruction**
  - If relation $R$ is decomposed into fragments $R_1, R_2, ..., R_n$, then there should exist some relational operator $\bigvee$ such that
    \[ R = \bigvee_{1 \leq i \leq n} R_i \]

- **Disjointness**
  - If relation $R$ is decomposed into fragments $R_1, R_2, ..., R_n$, and data item $d_i$ is in $R_j$, then $d_i$ should not be in any other fragment $R_k$ ($k \neq j$).

Allocation Alternatives

- **Non-replicated**
  - partitioned: each fragment resides at only one site

- **Replicated**
  - fully replicated: each fragment at each site
  - partially replicated: each fragment at some of the sites

- **Rule of thumb:**
  \[
  \frac{\text{read - only queries}}{\text{update queries}} \geq 1 \quad \text{replication is advantageous,}
  \]
  otherwise replication may cause problems
Comparison of Replication Alternatives

<table>
<thead>
<tr>
<th></th>
<th>Full-replication</th>
<th>Partial-replication</th>
<th>Partitioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY PROCESSING</td>
<td>Easy</td>
<td>Same Difficulty</td>
<td></td>
</tr>
<tr>
<td>DIRECTORY MANAGEMENT</td>
<td>Easy or Non-existant</td>
<td>Same Difficulty</td>
<td></td>
</tr>
<tr>
<td>CONCURRENCY CONTROL</td>
<td>Moderate</td>
<td>Difficult</td>
<td>Easy</td>
</tr>
<tr>
<td>RELIABILITY</td>
<td>Very high</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>REALITY</td>
<td>Possible application</td>
<td>Realistic</td>
<td>Possible application</td>
</tr>
</tbody>
</table>

Information Requirements

- Four categories:
  - Database information
  - Application information
  - Communication network information
  - Computer system information
Fragmentation

- Horizontal Fragmentation (HF)
  - Primary Horizontal Fragmentation (PHF)
  - Derived Horizontal Fragmentation (DHF)
- Vertical Fragmentation (VF)
- Hybrid Fragmentation (HF)

PHF – Information Requirements

- Database Information
  - relationship

  ![Database Diagram]

  - cardinality of each relation: \( \text{card}(R) \)
PHF - Information Requirements

**Application Information**

- **simple predicates**: Given $R[A_1, A_2, ..., A_n]$, a simple predicate $p_j$ is
  
  $p_j : A_i \theta \text{ Value}$

  where $\theta \in \{=, \leq, \geq, \neq\}$, $\text{Value} \in D_i$ and $D_i$ is the domain of $A_i$.

  For relation $R$ we define $Pr = \{p_1, p_2, ..., p_m\}$

  Example:
  
  $PNAME = \text{"Maintenance"}$

  $\text{BUDGET} \leq 200000$

- **minterm predicates**: Given $R$ and $Pr=\{p_1, p_2, ..., p_m\}$

  define $M=\{m_1, m_2, ..., m_r\}$ as

  $M=\{m_i \mid m_i = \wedge_{p_j \in Pr} p_j^*, 1 \leq j \leq m, 1 \leq i \leq z\}$

  where $p_j^* = p_j$ or $p_j^* = \neg(p_j)$.

Example

$m_1$: $PNAME = \text{"Maintenance"} \wedge \text{BUDGET} \leq 200000$

$m_2$: $\neg(PNAME = \text{"Maintenance"}) \wedge \text{BUDGET} \leq 200000$

$m_3$: $PNAME = \text{"Maintenance"} \wedge \neg(\text{BUDGET} \leq 200000)$

$m_4$: $\neg(PNAME = \text{"Maintenance"}) \wedge \neg(\text{BUDGET} \leq 200000)$
PHF – Information Requirements

- Application Information
  - minterm selectivities: \( \text{sel}(m_i) \)
    - The number of tuples of the relation that would be accessed by a user query which is specified according to a given minterm predicate \( m_i \).
  - access frequencies: \( \text{acc}(q_i) \)
    - The frequency with which a user application \( q_i \) accesses data.
    - Access frequency for a minterm predicate can also be defined.

Primary Horizontal Fragmentation

Definition:

\[
R_j = \sigma_{F_j}(R), \quad 1 \leq j \leq w
\]

where \( F_j \) is a selection formula, which is (preferably) a minterm predicate.

Therefore,

A horizontal fragment \( R_j \) of relation \( R \) consists of all the tuples of \( R \) which satisfy a minterm predicate \( m_i \).

\[
\downarrow
\]

Given a set of minterm predicates \( M \), there are as many horizontal fragments of relation \( R \) as there are minterm predicates.

Set of horizontal fragments also referred to as minterm fragments.
**PHF – Algorithm**

Given: A relation $R$, the set of simple predicates $Pr$
Output: The set of fragments of $R = \{R_1, R_2, \ldots, R_w\}$ which obey the fragmentation rules.

Preliminaries:
- $Pr$ should be complete
- $Pr$ should be minimal

**Completeness of Simple Predicates**

- A set of simple predicates $Pr$ is said to be complete if and only if the accesses to the tuples of the minterm fragments defined on $Pr$ requires that two tuples of the same minterm fragment have the same probability of being accessed by any application.

- Example:
  - Assume PROJ[PNO,PNAME,BUDGET,LOC] has two applications defined on it.
  - Find the budgets of projects at each location. (1)
  - Find projects with budgets less than $200000$. (2)
Completeness of Simple Predicates

According to (1),

\[ Pr = \{ \text{LOC} = \text{"Montreal"}, \text{LOC} = \text{"New York"}, \text{LOC} = \text{"Paris"} \} \]

which is not complete with respect to (2).

Modify

\[ Pr = \{ \text{LOC} = \text{"Montreal"}, \text{LOC} = \text{"New York"}, \text{LOC} = \text{"Paris"}, \text{BUDGET} \leq 200000, \text{BUDGET} > 200000 \} \]

which is complete.

Minimality of Simple Predicates

- If a predicate influences how fragmentation is performed, (i.e., causes a fragment \( f \) to be further fragmented into, say, \( f_i \) and \( f_j \)) then there should be at least one application that accesses \( f_i \) and \( f_j \) differently.
- In other words, the simple predicate should be relevant in determining a fragmentation.
- If all the predicates of a set \( Pr \) are relevant, then \( Pr \) is minimal.

\[
\frac{\text{acc}(m_i)}{\text{card}(f_i)} \neq \frac{\text{acc}(m_j)}{\text{card}(f_j)}
\]
Minimality of Simple Predicates

Example:

\[ Pr = \{ \text{LOC} = \text{Montreal}, \text{LOC} = \text{New York}, \text{LOC} = \text{Paris}, \text{BUDGET} \leq 200000, \text{BUDGET} > 200000 \} \]

is minimal (in addition to being complete).
However, if we add

\[ \text{PNAME} = \text{"Instrumentation"} \]

then \( Pr \) is not minimal.

---

**COM\_MIN Algorithm**

Given: a relation \( R \) and a set of simple predicates \( Pr \)

Output: a complete and minimal set of simple predicates \( Pr' \) for \( Pr \)

*Rule 1:* a relation or fragment is partitioned into at least two parts which are accessed differently by at least one application.
**COM_MIN Algorithm**

1. **Initialization**:
   - find a \( p_i \in Pr \) such that \( p_i \) partitions \( R \) according to Rule 1
   - set \( Pr' = p_i \); \( Pr \leftarrow Pr - p_i \); \( F \leftarrow f_i \)

2. **Iteratively add predicates to \( Pr' \) until it is complete**
   - find a \( p_j \in Pr \) such that \( p_j \) partitions some \( f_k \) defined according to minterm predicate over \( Pr' \) according to Rule 1
   - set \( Pr' = Pr' \cup p_j \); \( Pr \leftarrow Pr - p_j \); \( F \leftarrow F \cup f_i \)
   - if \( \exists p_k \in Pr' \) which is nonrelevant then
     - \( Pr' \leftarrow Pr' - p_k \)
     - \( F \leftarrow F - f_k \)

**PHORIzONTAL Algorithm**

Makes use of COM_MIN to perform fragmentation.

**Input:** a relation \( R \) and a set of simple predicates \( Pr \)

**Output:** a set of minterm predicates \( M \) according to which relation \( R \) is to be fragmented

1. \( Pr' \leftarrow \text{COM_MIN} (R, Pr) \)
2. determine the set \( M \) of minterm predicates
3. determine the set \( I \) of implications among \( p_i \in Pr \)
4. eliminate the contradictory minterms from \( M \)
**PHF – Example**

- Two candidate relations: PAY and PROJ.
- Fragmentation of relation PAY
  - Application: Check the salary info and determine raise.
  - Employee records kept at two sites ⇒ application run at two sites
  - Simple predicates
    - \( p_1: \text{SAL} \leq 30000 \)
    - \( p_2: \text{SAL} > 30000 \)
    - \( Pr = \{p_1, p_2\} \) which is complete and minimal \( Pr' = Pr \)
  - Minterm predicates
    - \( m_1: (\text{SAL} \leq 30000) \)
    - \( m_2: \text{NOT}(\text{SAL} \leq 30000) = (\text{SAL} > 30000) \)

---

**PHF – Example**

<table>
<thead>
<tr>
<th>PAY(_1)</th>
<th>PAY(_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TITLE</strong></td>
<td><strong>SAL</strong></td>
</tr>
<tr>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>Programmer</td>
<td>24000</td>
</tr>
<tr>
<td><strong>TITLE</strong></td>
<td><strong>SAL</strong></td>
</tr>
<tr>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
</tbody>
</table>
PHF – Example

[Fragmentation of relation PROJ]

Applications:
- Find the name and budget of projects given their no.
  - Issued at three sites
- Access project information according to budget
  - One site accesses ≤ 200000 other accesses > 200000

Simple predicates
- For application (1)
  - \( p_1 : \text{LOC} = \text{“Montreal”} \)
  - \( p_2 : \text{LOC} = \text{“New York”} \)
  - \( p_3 : \text{LOC} = \text{“Paris”} \)

- For application (2)
  - \( p_4 : \text{BUDGET} \leq 200000 \)
  - \( p_5 : \text{BUDGET} > 200000 \)

\[ Pr = Pr' = \{p_1, p_2, p_3, p_4, p_5\} \]

Minterm fragments left after elimination

- \( m_1 : (\text{LOC} = \text{“Montreal”}) \land (\text{BUDGET} \leq 200000) \)
- \( m_2 : (\text{LOC} = \text{“Montreal”}) \land (\text{BUDGET} > 200000) \)
- \( m_3 : (\text{LOC} = \text{“New York”}) \land (\text{BUDGET} \leq 200000) \)
- \( m_4 : (\text{LOC} = \text{“New York”}) \land (\text{BUDGET} > 200000) \)
- \( m_5 : (\text{LOC} = \text{“Paris”}) \land (\text{BUDGET} \leq 200000) \)
- \( m_6 : (\text{LOC} = \text{“Paris”}) \land (\text{BUDGET} > 200000) \)
## PHF – Example

<table>
<thead>
<tr>
<th>PROJ₁</th>
<th>PROJ₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNO</td>
<td>PNAME</td>
</tr>
<tr>
<td>P1</td>
<td>Instrumentation</td>
</tr>
<tr>
<td>P2</td>
<td>Database Develop.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJ₄</th>
<th>PROJ₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNO</td>
<td>PNAME</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
</tr>
</tbody>
</table>

## PHF – Correctness

- **Completeness**
  - Since $Pr'$ is complete and minimal, the selection predicates are complete.

- **Reconstruction**
  - If relation $R$ is fragmented into $F_R = \{R_1, R_2, \ldots, R_r\}$
    
    $$R = \bigcup_{i \in FR} R_i$$

- **Disjointness**
  - Mintern predicates that form the basis of fragmentation should be mutually exclusive.
**Derived Horizontal Fragmentation**

- Defined on a member relation of a link according to a selection operation specified on its owner.
  - Each link is an equijoin.
  - Equijoin can be implemented by means of semijoins.

![Diagram]

**DHF – Definition**

Given a link $L$ where $\text{owner}(L)=S$ and $\text{member}(L)=R$, the derived horizontal fragments of $R$ are defined as

$$R_i = R \bowtie_F S_i, \quad 1 \leq i \leq w$$

where $w$ is the maximum number of fragments that will be defined on $R$ and

$$S_i = \sigma_{F_i}(S)$$

where $F_i$ is the formula according to which the primary horizontal fragment $S_i$ is defined.
DHF – Example

Given link $L_1$ where $\text{owner}(L_1)=\text{SKILL}$ and $\text{member}(L_1)=\text{EMP}$

$$\text{EMP}_1 = \text{EMP} \bowtie \text{SKILL}_1$$
$$\text{EMP}_2 = \text{EMP} \bowtie \text{SKILL}_2$$

where

$$\text{SKILL}_1 = \sigma_{\text{SAL} \leq 30000}(\text{SKILL})$$
$$\text{SKILL}_2 = \sigma_{\text{SAL} > 30000}(\text{SKILL})$$

<table>
<thead>
<tr>
<th>EMP</th>
<th>ENO</th>
<th>ENAME</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4</td>
<td>J. Miller</td>
<td>Programmer</td>
<td></td>
</tr>
</tbody>
</table>

DHF – Correctness

- Completeness
  - Referential integrity
  - Let $R$ be the member relation of a link whose owner is relation $S$ which is fragmented as $F_S = \{S_1, S_2, ..., S_n\}$. Furthermore, let $A$ be the join attribute between $R$ and $S$. Then, for each tuple $t$ of $R$, there should be a tuple $t'$ of $S$ such that
    $$t[A]=t'[A]$$

- Reconstruction
  - Same as primary horizontal fragmentation.

- Disjointness
  - Simple join graphs between the owner and the member fragments.
Vertical Fragmentation

- Has been studied within the centralized context
  - design methodology
  - physical clustering
- More difficult than horizontal, because more alternatives exist.
  Two approaches:
  - grouping
    - attributes to fragments
  - splitting
    - relation to fragments

Vertical Fragmentation

- Overlapping fragments
  - grouping
- Non-overlapping fragments
  - splitting

We do not consider the replicated key attributes to be overlapping.

Advantage:

   Easier to enforce functional dependencies
   (for integrity checking etc.)
VF – Information Requirements

Application Information
- Attribute affinities
  - a measure that indicates how closely related the attributes are
  - This is obtained from more primitive usage data
- Attribute usage values
  - Given a set of queries \( Q = \{q_1, q_2, \ldots, q_n\} \) that will run on the relation \( R[A_1, A_2, \ldots, A_n] \),

\[
use(q_i, A_j) = \begin{cases} 
1 & \text{if attribute } A_j \text{ is referenced by query } q_i \\
0 & \text{otherwise}
\end{cases}
\]

\( use(q_i, *) \) can be defined accordingly.

VF – Definition of \( use(q_i, A_j) \)

Consider the following 4 queries for relation PROJ

\[
\begin{align*}
q_1: & \quad \text{SELECT} \quad \text{BUDGET} \\
& \quad \text{FROM} \quad \text{PROJ} \\
& \quad \text{WHERE} \quad \text{PNO=Value}
\end{align*}
\]

\[
\begin{align*}
q_2: & \quad \text{SELECT} \quad \text{PNAME, BUDGET} \\
& \quad \text{FROM} \quad \text{PROJ}
\end{align*}
\]

\[
\begin{align*}
q_3: & \quad \text{SELECT} \quad \text{PNAME} \\
& \quad \text{FROM} \quad \text{PROJ} \\
& \quad \text{WHERE} \quad \text{LOC=Value}
\end{align*}
\]

\[
\begin{align*}
q_4: & \quad \text{SELECT} \quad \text{SUM(BUDGET)} \\
& \quad \text{FROM} \quad \text{PROJ} \\
& \quad \text{WHERE} \quad \text{LOC=Value}
\end{align*}
\]

Let \( A_1 = \text{PNO}, A_2 = \text{PNAME}, A_3 = \text{BUDGET}, A_4 = \text{LOC} \)

\[
\begin{bmatrix}
A_1 & A_2 & A_3 & A_4 \\
q_1 & 1 & 0 & 1 & 0 \\
q_2 & 0 & 1 & 1 & 0 \\
q_3 & 0 & 1 & 0 & 1 \\
q_4 & 0 & 0 & 1 & 1
\end{bmatrix}
\]
VF – Affinity Measure $aff(A_i, A_j)$

The attribute affinity measure between two attributes $A_i$ and $A_j$ of a relation $R[A_1, A_2, \ldots, A_n]$ with respect to the set of applications $Q = (q_1, q_2, \ldots, q_q)$ is defined as follows:

$$
aff(A_i, A_j) = \sum_{\text{all queries that access } A_i \text{ and } A_j} \text{(query access)}
$$

$$
\text{query access} = \sum_{\text{all sites}} \text{access frequency of a query} \times \frac{\text{access execution}}{\text{all sites}}
$$

Assume each query in the previous example accesses the attributes once during each execution.

Also assume the access frequencies

$q_1$ = \begin{bmatrix} 15 & 20 & 10 \end{bmatrix}
$q_2$ = \begin{bmatrix} 5 & 0 & 0 \end{bmatrix}
$q_3$ = \begin{bmatrix} 25 & 25 & 25 \end{bmatrix}
$q_4$ = \begin{bmatrix} 3 & 0 & 0 \end{bmatrix}

Then

$$
aff(A_1, A_3) = 15 \times 1 + 20 \times 1 + 10 \times 1 = 45
$$

and the attribute affinity matrix $AA$ is

$$
AA = \begin{bmatrix} 45 & 0 & 45 & 0 \\
0 & 80 & 5 & 75 \\
45 & 5 & 53 & 3 \\
0 & 75 & 3 & 78 \end{bmatrix}
$$
VF – Clustering Algorithm

- Take the attribute affinity matrix $AA$ and reorganize the attribute orders to form clusters where the attributes in each cluster demonstrate high affinity to one another.
- Bond Energy Algorithm (BEA) has been used for clustering of entities. BEA finds an ordering of entities (in our case attributes) such that the global affinity measure

$$AM = \sum_i \sum_j (\text{affinity of } A_i \text{ and } A_j \text{ with their neighbors})$$

is maximized.

Bond Energy Algorithm

Input: The $AA$ matrix
Output: The clustered affinity matrix $CA$ which is a perturbation of $AA$

1. Initialization: Place and fix one of the columns of $AA$ in $CA$.
2. Iteration: Place the remaining $n-i$ columns in the remaining $i+1$ positions in the $CA$ matrix. For each column, choose the placement that makes the most contribution to the global affinity measure.
3. Row order: Order the rows according to the column ordering.
Bond Energy Algorithm

“Best” placement? Define contribution of a placement:

$$\text{cont}(A_i, A_h, A_j) = 2\text{bond}(A_i, A_h) + 2\text{bond}(A_h, A_i) - 2\text{bond}(A_i, A_j)$$

where

$$\text{bond}(A_x, A_y) = \sum_{z=1}^{kn} \text{aff}(A_z, A_x) \text{aff}(A_z, A_y)$$

BEA – Example

Consider the following AA matrix and the corresponding CA matrix where A1 and A2 have been placed. Place A3:

$$\text{AA} = \begin{bmatrix}
A_1 & A_2 & A_3 & A_4 \\
A_1 & 45 & 0 & 5 & 0 \\
A_2 & 0 & 80 & 5 & 75 \\
A_3 & 45 & 5 & 53 & 3 \\
A_4 & 0 & 75 & 3 & 78
\end{bmatrix}
\quad \text{CA} = \begin{bmatrix}
A_1 & A_2 \\
A_1 & 45 & 0 \\
A_2 & 0 & 80 \\
A_3 & 45 & 5 \\
A_4 & 0 & 75
\end{bmatrix}$$

Ordering (0-3-1):

$$\text{cont}(A_0, A_3, A_1) = 2\text{bond}(A_0, A_3) + 2\text{bond}(A_3, A_1) - 2\text{bond}(A_0, A_1)$$

$$= 2 \times 0 + 2 \times 4410 - 2 \times 0 = 8820$$

Ordering (1-3-2):

$$\text{cont}(A_1, A_3, A_2) = 2\text{bond}(A_1, A_3) + 2\text{bond}(A_3, A_2) - 2\text{bond}(A_1, A_2)$$

$$= 2 \times 4410 + 2 \times 890 - 2 \times 225 = 10150$$

Ordering (2-3-4):

$$\text{cont}(A_2, A_3, A_4) = 1780$$
Therefore, the CA matrix has to form

\[
\begin{bmatrix}
A_1 & A_3 & A_2 \\
45 & 45 & 0 \\
0 & 5 & 80 \\
45 & 53 & 5 \\
0 & 3 & 75 \\
\end{bmatrix}
\]

When \( A_4 \) is placed, the final form of the CA matrix (after row organization) is

\[
\begin{bmatrix}
A_1 & A_3 & A_2 & A_4 \\
A_1 & 45 & 45 & 0 & 0 \\
A_3 & 45 & 53 & 5 & 3 \\
A_2 & 0 & 5 & 80 & 75 \\
A_4 & 0 & 3 & 75 & 78 \\
\end{bmatrix}
\]
**VF – Algorithm**

How can you divide a set of clustered attributes \( \{A_1, A_2, \ldots, A_n\} \) into two (or more) sets \( \{A_1, A_2, \ldots, A_i\} \) and \( \{A_i, \ldots, A_n\} \) such that there are no (or minimal) applications that access both (or more than one) of the sets.

![Diagram](image)

**VF – Algorithm**

Define

- \( TQ \) = set of applications that access only \( TA \)
- \( BQ \) = set of applications that access only \( BA \)
- \( OQ \) = set of applications that access both \( TA \) and \( BA \)

and

- \( CTQ = \) total number of accesses to attributes by applications that access only \( TA \)
- \( CBQ = \) total number of accesses to attributes by applications that access only \( BA \)
- \( COQ = \) total number of accesses to attributes by applications that access both \( TA \) and \( BA \)

Then find the point along the diagonal that maximizes

\[ CTQ \times CBQ – COQ^2 \]
VF – Algorithm

Two problems:

1. Cluster forming in the middle of the CA matrix
   - Shift a row up and a column left and apply the algorithm to find the “best” partitioning point
   - Do this for all possible shifts
   - Cost $O(m^2)$

2. More than two clusters
   - $m$-way partitioning
   - Try $1, 2, \ldots, m-1$ split points along diagonal and try to find the best point for each of these
   - Cost $O(2^m)$

VF – Correctness

A relation $R$, defined over attribute set $A$ and key $K$, generates the vertical partitioning $F_R = \{R_1, R_2, \ldots, R_r\}$.

- Completeness
  - The following should be true for $A$:
    $$A = \bigcup A_{R_i}$$

- Reconstruction
  - Reconstruction can be achieved by
    $$R = \bowtie_{R_i} \forall R_i \in F_R$$

- Disjointness
  - TID’s are not considered to be overlapping since they are maintained by the system
  - Duplicated keys are not considered to be overlapping
Hybrid Fragmentation

Problem Statement

Given

\[ F = \{ F_1, F_2, ..., F_n \} \] fragments
\[ S = \{ S_1, S_2, ..., S_m \} \] network sites
\[ Q = \{ q_1, q_2, ..., q_q \} \] applications

Find the "optimal" distribution of \( F \) to \( S \).

Optimality

- Minimal cost
  - Communication + storage + processing (read & update)
  - Cost in terms of time (usually)
- Performance
  - Response time and/or throughput
- Constraints
  - Per site constraints (storage & processing)
Information Requirements

- Database information
  - selectivity of fragments
  - size of a fragment

- Application information
  - access types and numbers
  - access localities

- Communication network information
  - unit cost of storing data at a site
  - unit cost of processing at a site

- Computer system information
  - bandwidth
  - latency
  - communication overhead

Allocation

File Allocation (FAP) vs Database Allocation (DAP):
- Fragments are not individual files
  - relationships have to be maintained
- Access to databases is more complicated
  - remote file access model not applicable
  - relationship between allocation and query processing
- Cost of integrity enforcement should be considered
- Cost of concurrency control should be considered
Allocation – Information Requirements

- Database Information
  - selectivity of fragments
  - size of a fragment
- Application Information
  - number of read accesses of a query to a fragment
  - number of update accesses of query to a fragment
  - A matrix indicating which queries updates which fragments
  - A similar matrix for retrievals
  - originating site of each query
- Site Information
  - unit cost of storing data at a site
  - unit cost of processing at a site
- Network Information
  - communication cost/frame between two sites
  - frame size

Allocation Model

General Form

\[ \text{min}(\text{Total Cost}) \]

subject to

- response time constraint
- storage constraint
- processing constraint

Decision Variable

\[ x_{ij} = \begin{cases} 1 & \text{if fragment } F_i \text{ is stored at site } S_j \\ 0 & \text{otherwise} \end{cases} \]
**Allocation Model**

- **Total Cost**

\[
\sum_{\text{all queries}} \text{query processing cost} + \sum_{\text{all sites}} \sum_{\text{all fragments}} \text{cost of storing a fragment at a site}
\]

- **Storage Cost** (of fragment \(F_j\) at \(S_k\))

\[(\text{unit storage cost at } S_k) \times (\text{size of } F_j) \times x_{jk}\]

- **Query Processing Cost** (for one query)

  - processing component + transmission component

---

**Allocation Model**

- **Query Processing Cost**

  - **Processing component**
    
    access cost + integrity enforcement cost + concurrency control cost

  - **Access cost**

    \[
    \sum_{\text{all sites}} \sum_{\text{all fragments}} (\text{no. of update accesses} + \text{no. of read accesses}) \times x_{ij} \times \text{local processing cost at a site}
    \]

  - **Integrity enforcement and concurrency control costs**

    - Can be similarly calculated
**Allocation Model**

- **Query Processing Cost**
  
  **Transmission component**
  
  cost of processing updates + cost of processing retrievals
  
  → **Cost of updates**
  
  \[ \sum_{\text{all sites}} \sum_{\text{all fragments}} \text{update message cost} + \sum_{\text{all sites}} \sum_{\text{all fragments}} \text{acknowledgment cost} \]
  
  → **Retrieval Cost**
  
  \[ \sum_{\text{all fragments}} \min_{\text{all sites}} (\text{cost of retrieval command} + \text{cost of sending back the result}) \]

**Constraints**

- **Response Time**
  
  execution time of query ≤ max. allowable response time for that query
  
  → **Storage Constraint (for a site)**
  
  \[ \sum_{\text{all fragments}} \text{storage requirement of a fragment at that site} \leq \text{storage capacity at that site} \]
  
  → **Processing constraint (for a site)**
  
  \[ \sum_{\text{all queries}} \text{processing load of a query at that site} \leq \text{processing capacity of that site} \]
Allocation Model

- **Solution Methods**
  - FAP is NP-complete
  - DAP also NP-complete

- **Heuristics based on**
  - single commodity warehouse location (for FAP)
  - knapsack problem
  - branch and bound techniques
  - network flow

---

Allocation Model

- **Attempts to reduce the solution space**
  - assume all candidate partitionings known; select the “best” partitioning
  - ignore replication at first
  - sliding window on fragments