Outline

- Introduction & architectural issues
  - Data distribution
    - Fragmentation
    - Data Allocation
  - Distributed query processing
  - Distributed query optimization
  - Querying multidatabase systems
  - Distributed transactions & concurrency control
  - Distributed reliability
  - Database replication
  - Parallel database systems
  - Database integration & querying
  - Advanced topics

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
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

- Requirements Analysis
- Objectives
- User Input
- View Integration
- Conceptual Design
- View Design
- GCS
- Access Information
- ES's
- Distribution Design
- LCS's
- Physical Design
- LIS's
Distribution Design Issues

1. Why fragment at all?
2. How to fragment?
3. How much to fragment?
4. How to test correctness?
5. How to allocate?
6. 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

PROJ₁: projects with budgets less than $200,000

PROJ₂: 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 Develop.</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

PROJ₂: information about project names and locations

<table>
<thead>
<tr>
<th>PNO</th>
<th>BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>150000</td>
</tr>
<tr>
<td>P2</td>
<td>135000</td>
</tr>
<tr>
<td>P3</td>
<td>250000</td>
</tr>
<tr>
<td>P4</td>
<td>310000</td>
</tr>
<tr>
<td>P5</td>
<td>500000</td>
</tr>
</tbody>
</table>

<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

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 $\nabla$ such that
    $$ R = \nabla_{1\leq i\leq n} R_i $$

- Disjointness
  - If relation $R$ is decomposed into fragments $R_1$, $R_2$, ..., $R_n$, and data item $d_i$ is in $R_i$, 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:

\[
\text{If } \frac{\text{read-only queries}}{\text{update queries}} \geq 1 \text{ replication is advantageous, otherwise replication may cause problems}
\]

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

  ```
  PAY
  \[
  \begin{aligned}
  &\text{TITLE, SAL} \\
  \end{aligned}
  \]

  EMP
  \[
  \begin{aligned}
  &\text{ENO, ENAME, TITLE} \\
  \end{aligned}
  \]

  PROJ
  \[
  \begin{aligned}
  &\text{PNO, PNAME, BUDGET, LOC} \\
  \end{aligned}
  \]

  ASG
  \[
  \begin{aligned}
  &\text{ENO, PNO, RESP, DUR} \\
  \end{aligned}
  \]

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

- **Application Information**
  - **simple predicates**: Given \( R[A_1, A_2, \ldots, A_n] \), 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 \( Pr = \{p_1, p_2, \ldots, p_m\} \)
    Example:
    \[
    \text{PNAME} = \text{“Maintenance”} \quad \text{BUDGET} \leq 200000
    \]
  - **minterm predicates**: Given \( R \) and \( Pr = \{p_1, p_2, \ldots, p_m\} \)
    define \( M = \{m_1, m_2, \ldots, m_r\} \) as
    \[
    M = \{ m_i | m_i = \bigwedge_{p_j \in Pr} p_j^* \}, \ 1 \leq i \leq r
    \]
    where \( p_j^* = p_j \) or \( p_j^* = \neg(p_j) \).

**PHF – Minterm Examples**

- **Simple predicates on PROJ (partial)**
  - \( p_1 \): LOC = “Montreal”
  - \( p_2 \): LOC = “New York”
  - \( p_3 \): LOC = “Paris”
  - \( p_4 \): BUDGET \leq 200000
  - \( p_5 \): BUDGET \leq 200000

- **Minterm predicates on PROJECT (Partial)**
  - \( m_1 \): LOC = "Montreal" \( \land \) BUDGET \leq 200000
  - \( m_2 \): \( \neg(\text{LOC} = "\text{Montreal}"") \) \( \land \) BUDGET \leq 200000
  - \( m_3 \): LOC = "Montreal" \( \land \) \( \neg(\text{BUDGET} \leq 200000) \)
  - \( m_4 \): \( \neg(\text{LOC} = "\text{Montreal}"") \) \( \land \) \( \neg(\text{BUDGET} \leq 200000) \)
PHF – Information Requirements

- Application information.
  - minterm selectivities: \(sel(m_i)\).
    - 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: \(acc(q_i)\).
    - 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 \(m_i\).

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 $R$, $F_R = \{R_1, R_2, \ldots, R_w\}$ that 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 or equal to $200000$. (2)
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.

---

Completeness 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.
  \[
  \frac{\text{acc}(m)}{\text{card}(f)} \neq \frac{\text{acc}(m)}{\text{card}(f)}
  \]

---

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.
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 $Pr' = p_i$ ; $Pr' = Pr - p_i$ ; $F \leftarrow f_i$

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 $Pr' = Pr' \cup p_j$ ; $Pr' = Pr - p_j$ ; $F \leftarrow F \cup f_j$
   - 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 PROJ
  - Applications:
    - Find the name and budget of projects given their location
      - 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)
    - \( p_4 \): \( \text{BUDGET} \leq 200000 \)
    - \( p_5 \): \( \text{BUDGET} > 200000 \)
  - \( Pr = Pr' = \{ p_1, p_2, p_3, p_4, p_5 \} \)

PHF – Example

- Fragmentation of relation PROJ continued
  - 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><strong>PNO</strong></td>
<td><strong>PNAME</strong></td>
</tr>
<tr>
<td>P1</td>
<td>Instrumentation</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
</tr>
</tbody>
</table>

**PHF – Correctness**

- **Completeness**
  - Since $P^r$ is complete and minimal, the selection predicates are complete.

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

- **Disjointness**
  - Minterm predicates that form the basis of fragmentation should be mutually exclusive.
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 \( R[A_1, A_2, \ldots, A_n]\),
      \[
      \text{use}(q_i, A_j) = \begin{cases} 
        1 & \text{if attribute } A_j \text{ is referenced by query } q_i \\
        0 & \text{otherwise}
      \end{cases}
      \]
      \( \text{use}(q_i, \ast) \) can be defined accordingly

VF – Definition of \( \text{use}(q_i, A_j) \)

Consider the following 4 queries for relation PROJ:

\[
\begin{align*}
q_1: & \text{SELECT BUDGET FROM PROJ WHERE PNO=Value} \\
q_2: & \text{SELECT PNAME,BUDGET FROM PROJ} \\
q_3: & \text{SELECT PNAME FROM PROJ WHERE LOC=Value} \\
q_4: & \text{SELECT SUM(BUDGET) FROM PROJ WHERE LOC=Value}
\end{align*}
\]

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

<table>
<thead>
<tr>
<th></th>
<th>A_1</th>
<th>A_2</th>
<th>A_3</th>
<th>A_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>q_2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>q_3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>q_4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
VF – Affinity Measure $\text{aff}(A_i, A_j)$

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:

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

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

VF – Calculation of $\text{aff}(A_i, A_j)$

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

Also assume the access frequencies

<table>
<thead>
<tr>
<th>Query</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>15</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>$q_2$</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$q_3$</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>$q_4$</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Then

$$\text{aff}(A_1, A_3) = 15 \cdot 1 + 20 \cdot 1 + 10 \cdot 1$$

$$= 45$$

and the attribute affinity matrix $AA$ is

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>45</td>
<td>0</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>$A_2$</td>
<td>0</td>
<td>80</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>$A_3$</td>
<td>45</td>
<td>5</td>
<td>53</td>
<td>3</td>
</tr>
<tr>
<td>$A_4$</td>
<td>0</td>
<td>75</td>
<td>3</td>
<td>78</td>
</tr>
</tbody>
</table>
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 is maximized.

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

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:

\[
\text{cont}(A_i, A_k, A_j) = 2 \text{bond}(A_i, A_k) + 2 \text{bond}(A_k, A_j) - 2 \text{bond}(A_i, A_j)
\]

where

\[
\text{bond}(A_x, A_y) = \sum_{z=1}^{n} \text{aff}(A_z, A_x) \cdot \text{aff}(A_z, A_y)
\]

---

**BEA – Example**

Consider the following AA matrix and the corresponding CA matrix where \(A_1\) and \(A_2\) have been placed. Place \(A_3\):

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

\[
\begin{bmatrix}
A_1 & A_2 \\
A_1 & 45 & 0 \\
A_2 & 0 & 80 \\
A_3 & 45 & 5 \\
A_4 & 0 & 75
\end{bmatrix}
\]

**Ordering (0-3-1):**

\[
\text{cont}(A_0, A_3, A_1) = 2 \text{bond}(A_0, A_3) + 2 \text{bond}(A_3, A_1) - 2 \text{bond}(A_0, A_1) = 2 \cdot 0 + 2 \cdot 4410 - 2 \cdot 0 = 8820
\]

**Ordering (1-3-2):**

\[
\text{cont}(A_1, A_3, A_2) = 2 \text{bond}(A_1, A_3) + 2 \text{bond}(A_3, A_2) - 2 \text{bond}(A_1, A_2) = 2 \cdot 4410 + 2 \cdot 890 - 2 \cdot 225 = 10150
\]

**Ordering (2-3-4):**

\[
\text{cont}(A_2, A_3, A_4) = 1780
\]
BEA – Example

- Therefore, the CA matrix has the form
  \[
  \begin{bmatrix}
  A_1 & A_3 & A_2 \\
  45 & 45 & 0 \\
  0 & 5 & 80 \\
  45 & 53 & 5 \\
  0 & 3 & 75 \\
  \end{bmatrix}
  \]

- When \( A_i \) 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.
VF – Algorithm

Define
\[ TQ = \text{set of applications that access only } TA \]
\[ BQ = \text{set of applications that access only } BA \]
\[ OQ = \text{set of applications that access both } TA \text{ and } BA \]
and
\[ CTQ = \text{total number of accesses to attributes by applications that access only } TA \]
\[ CBQ = \text{total number of accesses to attributes by applications that access only } BA \]
\[ COQ = \text{total number of accesses to attributes by applications that access both } TA \text{ and } BA \]
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, ..., \( m-1 \) 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
    \[
    R = \bowtie \prod_{i} R_i \quad \forall R_i \in F_R
    \]

- **Disjointness**
  - TID's are not considered to be overlapping since they are maintained by the system
  - Duplicated keys are not considered to be overlapping

Fragment Allocation

- **Problem Statement**
  - Given
    \[
    F = \{ F_1, F_2, \ldots, F_n \} \quad \text{fragments}
    \]
    \[
    S = \{ S_1, S_2, \ldots, S_m \} \quad \text{network sites}
    \]
    \[
    Q = \{ q_1, q_2, \ldots, q_q \} \quad \text{applications}
    \]
  - 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 Model

**General Form**

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

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

- **Query Processing Cost** (for one query)
  processing component + transmission component