DDB Design: Recommended References

- Oracle, Rel 8.03, Documentation
- CS501R, D. Ng Distributed Databases, Lecture Notes
  - http://www.cs.byu.edu/courses/cs501r.2/notes.html

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
  - static
  - dynamic
- Data
  - Level of sharing
  - Level of knowledge
  - Complete information
  - Partial information

### 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

1. Requirements Analysis
2. Objectives
3. Conceptual Design
   - User Input
   - View Integration
4. View Design
5. Distribution Design
   - Access Information
   - ES's
   - LCS's
    - LCS's
   - Physical Design
   - LIs's

### 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

<table>
<thead>
<tr>
<th>J</th>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>$1$</td>
<td>Instrumentation</td>
<td>$150000$</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>$2$</td>
<td>Database Develop</td>
<td>$150000$</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>$3$</td>
<td>CAD/CAM</td>
<td>$250000$</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>$4$</td>
<td>Maintenance</td>
<td>$310000$</td>
<td>Paris</td>
</tr>
<tr>
<td>J5</td>
<td>$5$</td>
<td>CAD/CAM</td>
<td>$500000$</td>
<td>Boston</td>
</tr>
</tbody>
</table>

- $J_1$: projects with budgets less than $200,000$
- $J_2$: projects with budgets greater than or equal to $200,000$

## Fragmentation Alternatives – Vertical

<table>
<thead>
<tr>
<th>J</th>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>$1$</td>
<td>Instrumentation</td>
<td>$150000$</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>$2$</td>
<td>Database Develop</td>
<td>$135000$</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>$3$</td>
<td>CAD/CAM</td>
<td>$250000$</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>$4$</td>
<td>Maintenance</td>
<td>$310000$</td>
<td>Paris</td>
</tr>
<tr>
<td>J5</td>
<td>$5$</td>
<td>CAD/CAM</td>
<td>$500000$</td>
<td>Boston</td>
</tr>
</tbody>
</table>

- $J_1$: information about project budgets
- $J_2$: information about project names and locations

## Degree of Fragmentation

- finite number of alternatives
- tuples or attributes

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 $\gamma$ such that $R = \gamma_{\psi_{\text{join}}} R_i$.

- **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:**
  - If \( \frac{\text{read-only queries}}{\text{update queries}} \geq 1 \), 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-existent</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>
Fragmentation

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

PHF – Information Requirements

- Application Information
  - simple predicates: Given $RA_1, A_2, \ldots, A_n$, a simple predicate $p_j$ is $p_j : A_j \theta \text{Value}$
    where $\theta \in \{=, <, \leq, >, \geq, \neq\}$, $\text{Value} \in D_j$ and $D_j$ is the domain of $A_j$.
    For relation $R$ we define $Pr=(p_1, p_2, \ldots, p_m)$.
  - Example: $JNAME="Maintenance"$ \text{ and } $\text{BUDGET}\leq 200000$
  - minterm predicates: Given $R$ and $Pr=(p_1, p_2, \ldots, p_m)$ define $M=(m_1, m_2, \ldots, m_z)$ as $M=(m_1, m_2 = \land_j m_j, 1 \leq x \leq m_z)$
    where $m_1 = p_j$ or $m_2 = \neg(p_j)$.

PHF – Information Requirements

- Database Information
  - relationship $S$
  - $E \leftarrow L_1$
    - $JNO, JNAME, TITLE$
    - $JNO, JNAME, BUDGET, LOC$
  - $G \leftarrow L_2$
    - $ENO, JNO, RESP, DUR$
  - $card R$

Example

- $m_1: JNAME="Maintenance" \land BUDGET\leq 200000$
- $m_2: \neg JNAME="Maintenance" \land BUDGET\leq 200000$
- $m_3: JNAME = "Maintenance" \land \neg \text{BUDGET}\leq 200000$
- $m_4: \neg JNAME = "Maintenance" \land \neg \text{BUDGET}\leq 200000$
PHF – Information Requirements

- Application Information
  - mintern 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 mintern predicate \( m_i \).
  - access frequencies: \( acc \ q_i \)
    - The frequency with which a user application \( q_i \) accesses data.
    - Access frequency for a mintern predicate can also be defined.

Primary Horizontal Fragmentation

Definition:

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

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

Therefore,

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

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

Set of horizontal fragments also referred to as mintern fragments.

PHF – Algorithm

Given: A relation \( R \), the set of simple predicates \( Pr \)
Output: The set of fragments of \( R = \{ R_1, R_2, ..., R_n \} \) 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 mintern fragments defined on \( Pr \) requires that two tuples of the same mintern fragment have the same probability of being accessed by any application.

Example:

- Assume \( \{JNO,JNAME,BUDGET,LOC\} \) 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, \( f_1 \) and \( f_2 \)) then there should be at least one application that accesses \( f_1 \) and \( f_2 \) 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.

\[
\frac{\text{acc}(m_i)}{\text{card}(f)} \neq \frac{\text{acc}(m_j)}{\text{card}(f)}
\]

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
\[ \text{Name} = \text{"Instrumentation"} \]
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 \( f_i \) defined according to minterm predicate over \( Pr' \) according to Rule 1.
   - Set \( Pr' = Pr' \cup p_j \)
   - \( Pr \leftarrow Pr - p_j \)
   - \( F \leftarrow F \cup f_j \)
   - 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 \( \mathbf{I} \) of implications among \( p_i \in Pr \)
4. Eliminate the contradictory minterms from \( M \)

---

**PHF – Example**

- Two candidate relations: \( S \) and \( J \).
- **Fragmentation of relation \( S \)**
  - Application: Check the salary info and determine raise.
  - Employee records kept at two sites \( \Rightarrow \) 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
    - \( m_i : (\text{SAL} \leq 30000) \land (\text{SAL} > 30000) \)
    - \( m_2 : (\text{SAL} \leq 30000) \land \neg(\text{SAL} > 30000) \)
    - \( m_3 : \neg(\text{SAL} \leq 30000) \land (\text{SAL} > 30000) \)
    - \( m_4 : \neg(\text{SAL} \leq 30000) \land \neg(\text{SAL} > 30000) \)

**PHF – Example**

- **Fragmentation of relation \( S \) (continued)**
  - Implications
    - \( h : (\text{SAL} \leq 30000) \Rightarrow \neg(\text{SAL} > 30000) \)
    - \( h : \neg(\text{SAL} \leq 30000) \Rightarrow (\text{SAL} > 30000) \)
    - \( h_1 : (\text{SAL} > 30000) \Rightarrow \neg(\text{SAL} \leq 30000) \)
    - \( h_2 : \neg(\text{SAL} > 30000) \Rightarrow (\text{SAL} \leq 30000) \)
  - \( m_i \) is contradictory to \( h_i \), \( m_i \) is contradictory to \( h_2 \).

<table>
<thead>
<tr>
<th>TITLE</th>
<th>SAL</th>
<th>TITLE</th>
<th>SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer</td>
<td>24000</td>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
</tbody>
</table>
**PHF – Example**

- **Fragmentation of relation J**
  - Applications:
    - Find the name and budget of projects given their number.
    - Issued at three sites.
    - Access project information according to budget.
    - One site accesses ≤200000 other accesses >200000.
  - Simple predicates:
    - For application (1):
      - $p_1$: LOC = 'Montreal'
      - $p_2$: LOC = 'New York'
      - $p_3$: LOC = 'Paris'
    - For application (2):
      - $p_4$: BUDGET ≤ 200000
      - $p_5$: BUDGET > 200000
    - $P_r = P_r = \{p_1, p_2, p_3, p_4, p_5\}$

**PHF – Example**

- **Fragmentation of relation J continued**
  - Minterm fragments left after elimination:
    - $m_1$: (LOC = "Montreal") \& (BUDGET ≤ 200000)
    - $m_2$: (LOC = "Montreal") \& (BUDGET > 200000)
    - $m_3$: (LOC = "New York") \& (BUDGET ≤ 200000)
    - $m_4$: (LOC = "New York") \& (BUDGET > 200000)
    - $m_5$: (LOC = "Paris") \& (BUDGET ≤ 200000)
    - $m_6$: (LOC = "Paris") \& (BUDGET > 200000)

**PHF – Correctness**

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

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

- **Disjointness**
  - Minterm 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.

\[ S = \text{TITLE, SAL} \]

\[ E = \text{ENO, ENAME, TITLE, JNO, JNAME, BUDGET, LOC} \]

\[ G = \text{ENO, JNO, RESP, DUR} \]

DHF – Definition

Given a link \( L \) where \( \text{owner}(L) = S \) and \( \text{member}(L) = E \), 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 \( \text{owner}(L_1) = S \) and \( \text{member}(L_1) = E \)

\[ E_1 = E \bowtie S_1 \]
\[ E_2 = E \bowtie S_2 \]

where

\[ S_1 = \sigma_{\text{SAL} = 30000}(S) \]
\[ S_2 = \sigma_{\text{SAL} = 30000}(S) \]

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

<table>
<thead>
<tr>
<th>ENO</th>
<th>ENAME</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>J. Doe</td>
<td>Elect. Eng.</td>
</tr>
<tr>
<td>E2</td>
<td>M. Smith</td>
<td>Syst. Anal.</td>
</tr>
<tr>
<td>E5</td>
<td>B. Casey</td>
<td>Syst. Anal.</td>
</tr>
<tr>
<td>E8</td>
<td>J. Jones</td>
<td>Syst. Anal.</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 \( F_2 = (S_1, S_2, ..., S_w) \). 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 \( \forall A[t] = t'[A] \)

- 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
- splitting
  - attributes to fragments
  - relation to fragments

**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 \( Q = \{q_1, q_2, ..., q_d\} \) that will run on the relation \( R[A_1, A_2, ..., A_n] \)

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

veectors \( use(q_i, Q) \) can be defined accordingly

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

Consider the following 4 queries for relation J

\( q_1: \) SELECT BUDGET FROM J WHERE JNO = Value
\( q_2: \) SELECT JNAME,BUDGET FROM J WHERE JNO = Value
\( q_3: \) SELECT JNAME FROM J WHERE LOC = Value
\( q_4: \) SELECT SUM(BUDGET) FROM J

Let \( A_1 = \) JNO, \( A_2 = \) JNAME, \( A_3 = \) BUDGET, \( A_4 = \) LOC

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

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

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

Query access $= \sum_{\text{all sites}} \text{access frequency of a query * access execution}$

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>Site</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>15</td>
<td>0</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>$q_2$</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>$q_3$</td>
<td>15</td>
<td>0</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>$q_4$</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Then:

$$\text{aff}(A_i, A_j) = 15^*1 + 20^*1 + 10^*1$$

$$= 45$$

and the attribute affinity matrix $AA$ is

$$A = \begin{bmatrix}
15 & 0 & 45 & 0 \\
0 & 80 & 5 & 75 \\
45 & 5 & 3 & 78 \\
0 & 75 & 3 & 78
\end{bmatrix}$$

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:

\[ \text{con}(A_a, A_b, A_c) = 2 \text{bond}(A_a, A_b) + 2 \text{bond}(A_b, A_c) - 2 \text{bond}(A_a, A_c) \]

where

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

---

**BEA – Example**

Consider the following \(AA\) matrix and the corresponding \(CA\) matrix where \(A_1\) and \(A_2\) have been placed. Place \(A_3\) and \(A_4\):

\[
\begin{align*}
\text{AA} &= \\
A_1 & \quad A_2 & A_3 & A_4 \\
A_1 & 45 & 5 & 0 & 0 \\
A_2 & 0 & 80 & 5 & 75 \\
A_3 & 45 & 5 & 53 & 3 \\
A_4 & 0 & 75 & 3 & 78
\end{align*}
\]

\[
\begin{align*}
\text{CA} &= \\
A_1 & A_2 \\
A_1 & 45 & 0 \\
A_2 & 0 & 80 \\
A_3 & 45 & 5 \\
A_4 & 0 & 75
\end{align*}
\]

Ordering (0-3-1):

\[ \text{con}(A_0, A_3, A_1) = 2 \text{bond}(A_0, A_3) + 2 \text{bond}(A_3, A_1) - 2 \text{bond}(A_0, A_1) = 2^3 \times 0 - 2^0 \times 4410 + 2^2 \times 75 = 8820 \]

Ordering (1-3-2):

\[ \text{con}(A_1, A_3, A_2) = 2 \text{bond}(A_1, A_3) + 2 \text{bond}(A_3, A_2) - 2 \text{bond}(A_1, A_2) = 2^3 \times 4410 + 2^2 \times 890 - 2^2 \times 225 = 10150 \]

Ordering (2-3-4):

\[ \text{con}(A_2, A_3, A_4) = 1780 \]

---

**BEA – Example**

Therefore, the \(CA\) matrix has to form

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

---

**BEA – Example**

When \(A_1\) is placed, the final form of the \(CA\) matrix (after row organization) is

\[
\begin{align*}
\text{CA} &= \\
A_1 & A_3 & A_2 & A_4 \\
A_1 & 45 & 5 & 0 & 0 \\
A_2 & 45 & 53 & 5 & 3 \\
A_3 & 0 & 5 & 80 & 75 \\
A_4 & 0 & 3 & 75 & 78
\end{align*}
\]

---
VF – Algorithm

How can you divide a set of clustered attributes $(A_1, A_2, ..., A_n)$ into two (or more) sets $(A_1, A_2, ..., A_k)$ and $(A_{k+1}, ..., A_n)$ such that there are no (or minimal) applications that access both (or more than one) of the sets.

\[
\begin{array}{c}
A_1
\end{array}
\begin{array}{c}
A_2
\end{array}
\begin{array}{c}
\vdots
\end{array}
\begin{array}{c}
A_{k-1}
\end{array}
\begin{array}{c}
A_k
\end{array}
\begin{array}{c}
A_{k+1}
\end{array}
\begin{array}{c}
\vdots
\end{array}
\begin{array}{c}
A_n
\end{array}
\]

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 $CTQ - CBQ - COQ$

VF – Correctness

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

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

- **Reconstruction**
  - Reconstruction can be achieved by:
    \[ R = \bigcup_{R_i \in F_R} \cap R_i \]

- **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**

- **Graph**: 
  - Node $R$ with two hybrid fragments (HF) $R_1$ and $R_2$. 
  - $R_1$ has two virtual fragments (VF): $R_{11}$ and $R_{12}$. 
  - $R_2$ has two virtual fragments: $R_{21}$ and $R_{22}$.

**Fragment Allocation**

- **Problem Statement**
  - Given: 
    - $F = (F_1, F_2, ..., F_i)$ fragments
    - $S = (S_1, S_2, ..., S_m)$ network sites
    - $Q = (q_1, q_2, ..., q_n)$ 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 a 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

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

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

- 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:

\[
\sum_{\text{all sites}} \sum_{\text{all fragments}} (\text{no. of update accesses} + \text{no. of read accesses}) \times x_{ij}
\]

- Integrity enforcement and concurrency control costs

Can be similarly calculated
Allocation Model

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

- Constraints
  - Response Time:
    \[
    \text{execution time of query} \leq \text{max. allowable response time for that query}
    \]
  - 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 is also NP-complete

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

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

DDB Design: What we have covered?
- Design Strategies
- Distribution Design Issues
- Fragmentation
- Allocation

What’s next? – Client-server databases
- Overview of Client/Server Systems
- Middleware
- Java & JDBC
- SOA