## Outline

- ■ Introduction
- ■ Background
- ■ Distributed DBMS Architecture
- ❏ Distributed Database Design
  - ➠ Fragmentation
  - ➠ Data Location
- ❏ Semantic Data Control
- ❏ Distributed Query Processing
- ❏ Distributed Transaction Management
- ❏ Parallel Database Systems
- ❏ Distributed Object DBMS
- ❏ Database Interoperability
- ❏ Current Issues

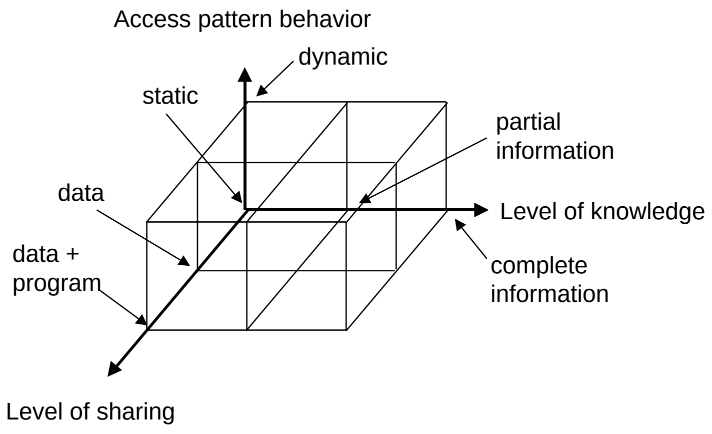
## Design Problem

- ■ 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

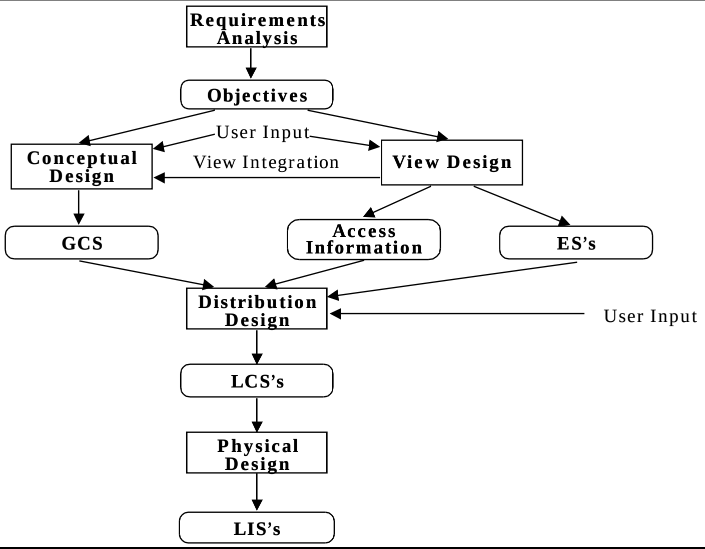
## Dimensions of the Problem



## 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

❶ Why fragment at all?

❷ How to fragment?

❸ How much to fragment?

❹ How to test correctness?

❺ How to allocate?

❻ 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

$PROJ_1$ : projects with budgets less than $200,000

$PROJ_2$ : projects with budgets greater than or equal to $200,000

PROJ

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P1 | Instrumentation | 150000 | Montreal |
| P2 | Database Develop. | 135000 | |
| P3 | CAD/CAM | 250000 | |
| P4 | Maintenance | 310000 | Paris |
| P5 | CAD/CAM | 500000 | Boston |

$PROJ_1$

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P1 | Instrumentation | 150000 | Montreal |
| P2 | Database Develop. | 135000 | New York |

$PROJ_2$

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P3 | CAD/CAM | 250000 | New York |
| P4 | Maintenance | 310000 | Paris |
| P5 | CAD/CAM | 500000 | Boston |

## Fragmentation Alternatives – Vertical

PROJ$_1$: information about project budgets

PROJ$_2$: information about project names and locations

PROJ

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P1 | Instrumentation | 150000 | Montreal |
| P2 | Database Develop. | 135000 | |
| P3 | CAD/CAM | 250000 | |
| P4 | Maintenance | 310000 | Paris |
| P5 | CAD/CAM | 500000 | Boston |

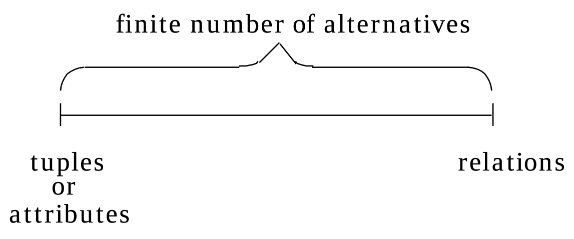
PROJ$_1$

| PNO | BUDGET |
|---|---|
| P1 | 150000 |
| P2 | 135000 |
| P3 | 250000 |
| P4 | 310000 |
| P5 | 500000 |

PROJ$_2$

| PNO | PNAME | LOC |
|---|---|---|
| P1 | Instrumentation | Montreal |
| P2 | Database Develop. | New York |
| P3 | CAD/CAM | New York |
| P4 | Maintenance | Paris |
| P5 | CAD/CAM | Boston |

## Degree of Fragmentation



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 such that

$$R = \ _{1 \le i \le n} R_i$$

- 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:

If $\frac{\text{read - only queries}}{\text{update quries}} \geq 1$ replication is advantageous,

otherwise replication may cause problems

## Comparison of Replication Alternatives

| | Full-replication | Partial-replication | Partitioning |
|---|---|---|---|
| QUERY PROCESSING | Easy | Same Difficulty | |
| DIRECTORY MANAGEMENT | Easy or Non-existant | Same Difficulty | |
| CONCURRENCY CONTROL | Moderate | Difficult | Easy |
| RELIABILITY | Very high | High | Low |
| REALITY | Possible application | Realistic | Possible application |

## Information Requirements

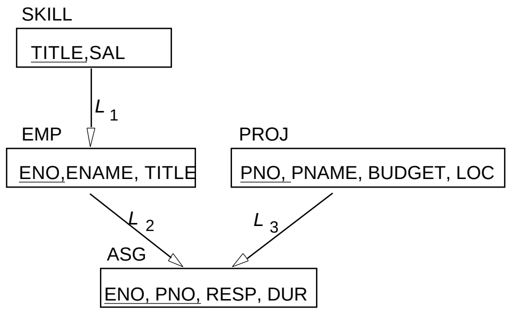
- Four categories:
  - Database information
  - Application information
  - Communication network information
  - Computer system information

## Fragmentation

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

## PHF – Information Requirements

- Database Information
  - relationship



  - cardinality of each relation: *card*(*R*)

## PHF - Information Requirements

- Application Information
  - **simple predicates** : Given $R[A_1, A_2, \ldots, A_n]$, a simple predicate $p_j$ is

    $p_j : A_i \,\theta\, Value$

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

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

    Example :

    PNAME = "Maintenance"

    BUDGET ≤ 200000
  - **minterm predicates** : Given $R$ and $Pr=\{p_1, p_2, \ldots, p_m\}$ define $M=\{m_1, m_2, \ldots, m_r\}$ as

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

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

## PHF – Information Requirements

Example

$m_1$: PNAME="Maintenance" BUDGET≤200000

$m_2$: **NOT**(PNAME="Maintenance") BUDGET≤200000

$m_3$: PNAME= "Maintenance" **NOT**(BUDGET≤200000)

$m_4$: **NOT**(PNAME="Maintenance") **NOT**(BUDGET≤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 *qi* 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 $R_i$ 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 $R = \{R_1, R_2,\ldots,R_w\}$ 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*={LOC="Montreal",LOC="New York",LOC="Paris"}

which is not complete with respect to (2).

Modify

*Pr* ={LOC="Montreal",LOC="New York",LOC="Paris",
BUDGET≤200000,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*.

$$\frac{acc(m_i)}{card(f_i)} \neq \frac{acc(m_j)}{card(f_j)}$$

## Minimality of Simple Predicates

Example :

*Pr* ={LOC="Montreal",LOC="New York", LOC="Paris", BUDGET≤200000,BUDGET>200000}

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

PNAME = "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

❶ Initialization :

- find a $p_i$   $Pr$ such that $p_i$ partitions $R$ according to *Rule 1*
- set $Pr' = p_i$ ; $Pr \leftarrow Pr - p_i$ ; $F \leftarrow f_i$

❷ Iteratively add predicates to $Pr'$ until it is complete

- find a $p_j$   $Pr$ such that $p_j$ partitions some $f_k$ defined according to minterm predicate over $Pr'$ according to *Rule 1*
- set $Pr' = Pr'$   $p_i$ ; $Pr \leftarrow Pr - p_i$; $F \leftarrow F$   $f_i$
- if   $p_k$   $Pr'$ 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

❶ $Pr' \leftarrow$ COM_MIN ($R$,$Pr$)

❷ determine the set $M$ of minterm predicates

❸ determine the set $I$ of implications among $p_i$   $Pr$

❹ 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

    $p_1$ : SAL ≤ 30000

    $p_2$ : SAL > 30000

    $Pr = \{p_1,p_2\}$ which is complete and minimal $Pr'=Pr$
  - Minterm predicates

    $m_1$ : (SAL ≤ 30000)

    $m_2$ : **NOT**(SAL ≤ 30000) = (SAL > 30000)

## PHF – Example

$PAY_1$

| TITLE | SAL |
|---|---|
| Mech. Eng. | 27000 |
| Programmer | 24000 |

$PAY_2$

| TITLE | SAL |
|---|---|
| Elect. Eng. | 40000 |
| Syst. Anal. | 34000 |

## 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)

    $p_1$ : LOC = “Montreal”

    $p_2$ : LOC = “New York”

    $p_3$ : LOC = “Paris”
  - For application (2)

    $p_4$ : BUDGET ≤ 200000

    $p_5$ : 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

    $m_1$ : (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 – Example

$PROJ_1$

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P1 | Instrumentation | 150000 | Montreal |

$PROJ_2$

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P2 | Database Develop. | 135000 | New York |

$PROJ_4$

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P3 | CAD/CAM | 250000 | New York |

$PROJ_6$

| PNO | PNAME | BUDGET | LOC |
|---|---|---|---|
| P4 | Maintenance | 310000 | Paris |

## PHF – Correctness

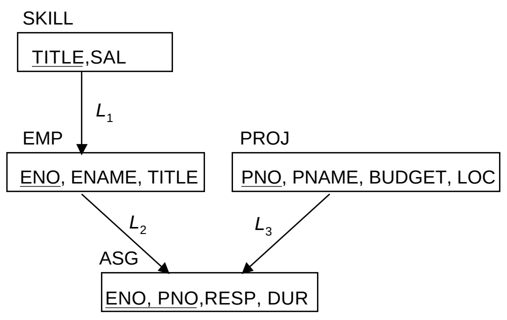
- Completeness
  - Since *Pr*' is complete and minimal, the selection predicates are complete
- Reconstruction
  - If relation *R* is fragmented into $F_R = \{R_1, R_2, \ldots, R_r\}$

$$R = {}_{R_i \quad FR} R_i$$

- 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.



## DHF – Definition

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

$$R_i = R \ltimes_F S_i,\ 1\leq i\leq w$$

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 owner($L_1$)=SKILL and member($L_1$)=EMP

$EMP_1 = EMP \ltimes SKILL_1$

$EMP_2 = EMP \ltimes SKILL_2$

where

$SKILL_1 = \sigma_{SAL \leq 30000}(SKILL)$

$SKILL_2 = \sigma_{SAL > 30000}(SKILL)$

$EMP_1$

| ENO | ENAME | TITLE |
|---|---|---|
| E3 | A. Lee | Mech. Eng. |
| E4 | J. Miller | Programmer |
| E7 | R. Davis | Mech. Eng. |

$EMP_2$

| ENO | ENAME | TITLE |
|---|---|---|
| E1 | J. Doe | Elect. Eng. |
| E2 | M. Smith | Syst. Anal. |
| E5 | B. Casey | Syst. Anal. |
| E6 | L. Chu | Elect. Eng. |
| E8 | J. Jones | Syst. Anal. |

## 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 $Q = \{q_1, q_2, \ldots, q_q\}$ that will run on the relation $R[A_1, A_2, \ldots, A_n]$,

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

$use(q_i, \bullet)$ can be defined accordingly

## VF – Definition of $use(q_i, A_j)$

Consider the following 4 queries for relation PROJ

| | | | | | |
|---|---|---|---|---|---|
| $q_1$: | **SELECT** | BUDGET | $q_2$: | **SELECT** | PNAME,BUDGET |
| | **FROM** | PROJ | | **FROM** | PROJ |
| | **WHERE** | PNO=Value | | | |
| $q_3$: | **SELECT** | PNAME | $q_4$: | **SELECT** | **SUM**(BUDGET) |
| | **FROM** | PROJ | | **FROM** | PROJ |
| | **WHERE** | LOC=Value | | **WHERE** | LOC=Value |

Let $A_1$= PNO, $A_2$= PNAME, $A_3$= BUDGET, $A_4$= LOC

$$\begin{array}{c|cccc|} & A_1 & A_2 & A_3 & A_4 \\ q_1 & 1 & 0 & 1 & 0 \\ q_2 & 0 & 1 & 1 & 0 \\ q_3 & 0 & 1 & 0 & 1 \\ q_4 & 0 & 0 & 1 & 1 \end{array}$$

## VF – Affinity Measure $aff(A_i,A_j)$

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 :

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

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

## VF – Calculation of $aff(A_i, A_j)$

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

Also assume the access frequencies $\Longrightarrow$

$$\begin{array}{c|ccc} & S_1 & S_2 & S_3 \\ \hline q_1 & 15 & 20 & 10 \\ q_2 & 5 & 0 & 0 \\ q_3 & 25 & 25 & 25 \\ q_4 & 3 & 0 & 0 \end{array}$$

Then

$aff(A_1, A_3)$ = 15*1 + 20*1+10*1

= 45

and the attribute affinity matrix $AA$ is $\Longrightarrow$

$$\begin{array}{c|cccc} & A_1 & A_2 & A_3 & A_4 \\ \hline A_1 & 45 & 0 & 45 & 0 \\ A_2 & 0 & 80 & 5 & 75 \\ A_3 & 45 & 5 & 53 & 3 \\ A_4 & 0 & 75 & 3 & 78 \end{array}$$

## 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 perturbationof *AA*

❶ *Initialization*: Place and fix one of the columns of *AA* in *CA*.

❷ *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.

❸ *Row order*:Order the rows according to the column ordering.

## Bond Energy Algorithm

"Best" placement? Define contribution of a placement:

$$cont(A_i, A_k, A_j) = 2bond(A_i, A_k) + 2bond(A_k, A_l) - 2bond(A_i, A_j)$$

where

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

## BEA – Example

Consider the following *AA* matrix and the corresponding *CA* matrix where $A_1$ and $A_2$ have been placed. Place $A_3$:

$$AA = \begin{array}{c|cccc|} & A_1 & A_2 & A_3 & A_4 \\ A_1 & 45 & 0 & 5 & 0 \\ A_2 & 0 & 80 & 5 & 75 \\ A_3 & 45 & 5 & 53 & 3 \\ A_4 & 0 & 75 & 3 & 78 \end{array} \qquad CA = \begin{array}{|cccc|} A_1 & A_2 & & \\ 45 & 0 & & \\ 0 & 80 & & \\ 45 & 5 & & \\ 0 & 75 & & \end{array}$$

Ordering (0-3-1) :

$cont(A_0,A_3,A_1) = 2bond(A_0 , A_3)+2bond(A_3 , A_1)–2bond(A_0 , A_1)$
$= 2* 0 + 2* 4410 – 2*0 = 8820$

Ordering (1-3-2) :

$cont(A_1,A_3,A_2) = 2bond(A_1 , A_3)+2bond(A_3 , A_2)–2bond(A_1,A_2)$
$= 2* 4410 + 2* 890 – 2*225 = 10150$

Ordering (2-3-4) :

$cont (A_2,A_3,A_4) = 1780$

## BEA – Example

Therefore, the *CA* matrix has to form

$$\begin{array}{c}
\begin{array}{ccc} A_1 & A_3 & A_2 \end{array} \\
\begin{bmatrix} 45 & 45 & 0 \\ 0 & 5 & 80 \\ 45 & 53 & 5 \\ 0 & 3 & 75 \end{bmatrix}
\end{array}$$

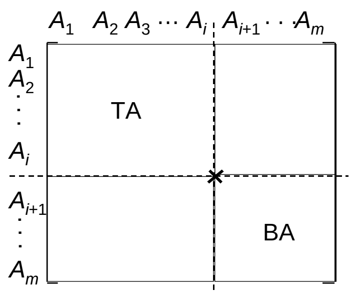
## BEA – Example

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

$$\begin{array}{c|cccc}
 & A_1 & A_3 & A_2 & A_4 \\
\hline
A_1 & 45 & 45 & 0 & 0 \\
A_3 & 45 & 53 & 5 & 3 \\
A_2 & 0 & 5 & 80 & 75 \\
A_4 & 0 & 3 & 75 & 78
\end{array}$$

## 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* = 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

$$CTQ \quad CBQ - COQ^2$$

## VF – Algorithm

Two problems :

❶ 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 $O(m^2)$

❷ 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 $A$:

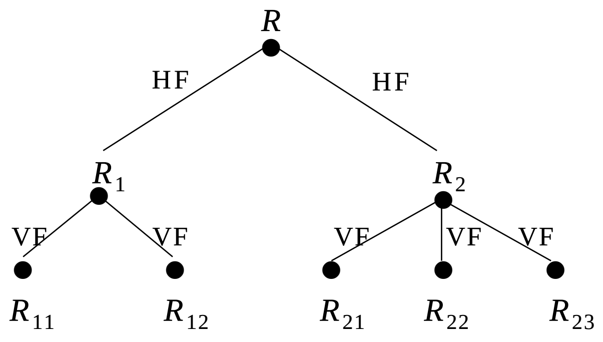
$$A = \quad A_{R_i}$$

- Reconstruction
  - Reconstruction can be achieved by

$$R = \bowtie_K R_i \quad R_i \quad F_R$$

- 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



## Fragment Allocation

- Problem Statement

  Given

| | |
|---|---|
| $F = \{F_1, F_2, \ldots, F_n\}$ | fragments |
| $S = \{S_1, S_2, \ldots, S_m\}$ | network sites |
| $Q = \{q_1, q_2, \ldots, q_q\}$ | 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**

min(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

  $\sum_{\text{all queries}}$ query processing cost +

  $\sum_{\text{all sites}} \sum_{\text{all fragments}}$ cost of storing a fragment at a site

- Storage Cost (of fragment $F_j$ at $S_k$)

  (unit storage cost at $S_k$) (size of $F_j$) $x_{jk}$

- 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

    $\sum_{\text{all sites}} \sum_{\text{all fragments}}$ (no. of update accesses+ no. of read accesses)

    $x_{ij}$ 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

  $\sum_{\text{all sites}} \sum_{\text{all fragments}}$ update message cost +

  $\sum_{\text{all sites}} \sum_{\text{all fragments}}$ acknowledgment cost

  ➠ Retrieval Cost

  $\sum_{\text{all fragments}} \min_{\text{all sites}}$(cost of retrieval command +

  cost of sending back the result)

## Allocation Model

- Constraints

  ➠ Response Time

  execution time of query ≤ max. allowable response time for that query

  ➠ Storage Constraint (for a site)

  $\sum_{\text{all fragments}}$ storage requirement of a fragment at that site ≤

  storage capacity at that site

  ➠ Processing constraint (for a site)

  $\sum_{\text{all queries}}$ processing load of a query at that site ≤

  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