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, this entails:
  - Placement of the distributed DBMS software; and
  - Placement of the applications that run on the database.
Distributed Design

- **Top-Down**
  - Mostly in designing systems from scratch
  - Mostly in homogeneous systems
- **Bottom-Up**
  - When the constituent databases already exist at a number of sites.

Distributed Design Issues

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

- What is a reasonable unit of distribution?
  - Relations
    - Views are subsets of relations → locality
    - Extra communication
  - Fragments of relations
    - Concurrent execution of a number of txns on the same relation
    - Views that cannot be defined on a single fragment will require extra processing
    - Semantic data control (especially integrity enforcement) more difficult

Types of fragmentation

- Horizontal
  - Divide tuples based upon certain properties, e.g. ranges.
- Vertical
  - Divide attributes
    - Need to replicate primary key attributes
- Hybrid
  - Alternating application of horizontal and vertical.
Correctness of fragmentation

- Completeness
  - Decomposition of Relation $R$ into $R_1, R_2, ... R_n$ 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 $R_1, R_2, ... R_n$, then there should exist some operator, that $R$ can be reconstructed from $R_1, ... R_n$.

- Disjointness
  - If Relation $R$ is decomposed into $R_1, R_2, ... R_n$, and data item $d$ is in $R_j$, then $d$ should not be in any other fragment $R_k$, $k <> j$.

Allocation Alternatives

- Non-replicated
  - Partitioned: each fragment resides at only one site

- Replicated
  - Fully replicated
  - Partially replicated

- Rule of thumb:
  - If (read-only queries/update queries) $\geq 1$ replication is advantageous
Comparison of alternatives

<table>
<thead>
<tr>
<th></th>
<th>Full Replication</th>
<th>Partial Replication</th>
<th>Partitioning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query Processing</strong></td>
<td>Easy</td>
<td>Same</td>
<td>Same</td>
</tr>
<tr>
<td><strong>Directory Management</strong></td>
<td>Easy or non-existant</td>
<td>Same</td>
<td>Same</td>
</tr>
<tr>
<td><strong>Concurrency Control</strong></td>
<td>Moderate</td>
<td>Difficult</td>
<td>Easy</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>Very High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Reality</strong></td>
<td>Possible Application</td>
<td>Realistic</td>
<td>Possible application</td>
</tr>
</tbody>
</table>

Information Requirements

- Four **categories of information** are required for distributed database design:
  - *Database Information*
  - *Application Information*
  - *Communication network information*
  - *Computer system information*
Horizontal Fragmentation

- There are two types:
  - Primary
    - Based upon values of attributes in the relation being fragmented
  - Derived
    - Based upon values of attributes of some other relation.

Primary Horizontal Fragmentation

- Database Information
  - Relationship
    - SKILL
      - Title, Sal
    - EMP
      - ENO, Ename, Title
    - ASG
      - ENO, PNO, Resp, Dur
    - PROJ
      - PNO, Pname, Budget, Loc
  - 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 \( \theta \) is a comparison operator, Value is from the domain of attribute \( A_i \)
  – **Minterm predicates**: Given \( R \) and \( P_r = \{p_1, p_2, \ldots, p_m\} \), define \( M = \{m_1, m_2, \ldots, m_z\} \) as
    \[
    M = \{m_i \mid m_i = \wedge_{p_j \in P_r} p_j^* \}, 1 \leq i \leq z
    \]
    where \( p_j^* = p_j \) or \( \text{NOT}(p_j) \).

PHF – Information Requirements

• Examples
  – PNAME = “Maintenance” AND BUDGET <= 200000
  – NOT(PNAME=“Maintenance”) AND BUDGET <= 200000
  – PNAME = “Maintenance” AND NOT(BUDGET <=200000)
  – NOT(PNAME=“Maintenance”) AND NOT(Budget<=200000)
PHF-Information Req.

• 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 query \( q_i \) is accessed
    • Access frequency of a minterm predicate can also be defined.

Primary Horizontal Frag.

• Definition: \( R_j = \sigma_{F_j}(R), 1 \leq j \leq w \)
  – 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 \( m_j \).
  – Given a minterm of 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 $P_r$
- **OUTPUT**: The set of fragments of $R = \{R_1, \ldots, R_w\}$ which obey the fragmentation rules.
- **Preliminaries**:
  - $P_r$ should be complete
  - $P_r$ should be minimal

### Completeness of Simple Predicates

- A set of simple predicates $P_r$ is said to be **complete** iff the accesses to the tuples of the minterm fragments defined on $P_r$ requires that two tuples of the same minterm fragment have the same probability of being accessed by the 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),
  - \( P_r = \{\text{LOC}="\text{Montreal}", \text{LOC}="\text{New York}", \text{LOC}="\text{Paris}\} \)
- Which is not complete with respect to (2).
- Modify
  - \( P_r = \{\text{LOC}="\text{Montreal}", \text{LOC}="\text{New York}", \text{LOC}="\text{Paris}\}, \text{BUDGET} \leq 200000, \text{BUDGET} > 200000 \} \)
- 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 \( P_r \) are relevant, then \( P_r \) is minimal. 
  \[
  \frac{\text{acc}(m_i)}{\text{card}(f_i)} \neq \frac{\text{acc}(m_j)}{\text{card}(f_j)}
  \]
COM-MIN Algorithm

• **Given:** a relation $R$ and a set of simple predicates $P_r$.
• **Output:** a complete and minimal set of simple predicates $P_r'$ for $P_r$.

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

PHORizontal Algorithm

• **Makes use of COM-MIN** to perform fragmentation.
• **Input:** a relation $R$ and a set of simple predicates $P_r$
• **Output:** a set of minterm predicates $M$ according to which $R$ is to be fragmented.

1. $P_r' \leftarrow \text{COM-MIN}(R, P_r)$
2. Determine the set $M$ of minterm predicates
3. Determine the set $I$ of implications among $p_i$ from $P_r$.
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
    - $p_1 : \text{SAL} \leq 30000$
    - $p_2 : \text{SAL} > 30000$
    - $P_r = \{p_1, p_2\}$ which is complete and minimal $P_r = P_r$
  - Minterm predicates
    - $m_1 : (\text{SAL} \leq 30000)$
    - $m_2 : \text{NOT}(\text{SAL} \leq 30000) = (\text{SAL} > 30000)$

<table>
<thead>
<tr>
<th>TITLE</th>
<th>SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>Programmer</td>
<td>24000</td>
</tr>
<tr>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
</tbody>
</table>
Fragmentation of 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 another 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 PROJ contd:**
  - Minterm fragments left after elimination
    - \( m_1 \): (LOC = "Montreal") AND (BUDGET <= 200000)
    - \( m_2 \): (LOC = "Montreal") AND (BUDGET > 200000)
    - \( m_3 \): (LOC = "New York") AND (BUDGET <= 200000)
    - \( m_4 \): (LOC = "New York") AND (BUDGET > 200000)
    - \( m_5 \): (LOC = "Paris") AND (BUDGET <= 200000)
    - \( m_6 \): (LOC = "Paris") AND (BUDGET > 200000)
PHF -- Example

<table>
<thead>
<tr>
<th>PROJ₁</th>
<th>PROJ₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNO</td>
<td>PNO</td>
</tr>
<tr>
<td>PNAME</td>
<td>PNAME</td>
</tr>
<tr>
<td>BUDGET</td>
<td>BUDGET</td>
</tr>
<tr>
<td>LOC</td>
<td>LOC</td>
</tr>
<tr>
<td>P1</td>
<td>P2</td>
</tr>
<tr>
<td>Instr.</td>
<td>Database Develop.</td>
</tr>
<tr>
<td>150000</td>
<td>135000</td>
</tr>
<tr>
<td>Montreal</td>
<td>New York</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJ₄</th>
<th>PROJ₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNO</td>
<td>PNO</td>
</tr>
<tr>
<td>PNAME</td>
<td>PNAME</td>
</tr>
<tr>
<td>BUDGET</td>
<td>BUDGET</td>
</tr>
<tr>
<td>LOC</td>
<td>LOC</td>
</tr>
<tr>
<td>P3</td>
<td>P4</td>
</tr>
<tr>
<td>CAD/CA M</td>
<td>Maint.</td>
</tr>
<tr>
<td>250000</td>
<td>310000</td>
</tr>
<tr>
<td>New York</td>
<td>Paris</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 $F_R=\{R_1, R_2, \ldots R_d\}$

• **Disjointness** $R = \bigcup_{\forall R_i \in F_R} R_i$
  – 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

\[ R_i = R \bowtie_{F_i} S_i, 1 \leq i \leq w \]

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 L1 where owner(L1) = SKILL and member(L1) = EMP

\[ EMP_1 = EMP \bowtie SKILL_1 \]
\[ EMP_2 = EMP \bowtie SKILL_2 \]

where

\[ SKILL_1 = \sigma_{SAL\leq30000}(SKILL) \]
\[ SKILL_2 = \sigma_{SAL>30000}(SKILL) \]
DHF – Example

<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 $Fs=\{S1, S2, \ldots, Sn\}$. 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 HF

- **Disjointness**
  - Simple join graphs between the owner and 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
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 \( Q = \{ q_1, q_2, ..., q_k \} \) that will run on the relation \( R[A_1, A_2, ..., A_n] \),
    - Use \((q_i, A_j) = 1 \) if \( A_j \) is referenced by \( q_i \), 0 otherwise
    - Use \((q_i, \_\_\_)\) can be defined accordingly

VF – Definition of use(qi,Aj)

- Consider the following 4 queries for PROJ
  
<table>
<thead>
<tr>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT BUDGET FROM PROJ WHERE PNO=Value</td>
<td>SELECT PNAME, BUDGET FROM PROJ</td>
<td>SELECT PNAME FROM PROJ WHERE LOC=Value</td>
<td>SELECT SUM(BUDGET) FROM PROJ WHERE LOC=Value</td>
</tr>
</tbody>
</table>

- Let \( A_1=PNO, A_2=PNAME, A_3=BUDGET, A_4=LOC \)

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

- The attribute affinity measure between two attributes $A_i$ and $A_j$ of a relation $R$ with respect to the set of applications $Q=\{q_1, q_2, \ldots, q_k\}$ is defined as follows:

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

$$\text{query access} = \sum_{\text{allsites}} \text{access freq of a query} \times \frac{\text{access}}{\text{execution}}$$

VF – Clustering Algorithm

- Take the attribute affinity matrix $AA$ and reorganize the attribute orders to form clusters where the attributes in each cluster have high affinity for each other.
- Bond Energy Algorithm (BEA) has been used for clustering of attributes. This algorithm finds clustering 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 (a perturbation of AA)

1. **Initialization**: Place and fix one of the columns of AA in CA
2. **Iteration**: Place the remaining n-1 columns in the remaining I+1 positions in the CA matrix. For each column, chose the placement that makes the most contribution to the global affinity measure.
3. **Row Order**: Order the rows according to the columns.

**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, \ldots, A_i\} and \{A_{i+1}, \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
  – CQ – set of applications that access both
• 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 TB
  – COQ – total number of accesses to attributes by applications that access both TA and TB
• Then find the point along the diagonal that maximizes $CTQ \times CBQ - COQ^2$

VF – Algorithms

• Two problems:
  1. Cluster forming in the middle of CA
     1. Shift a row up, and a column left and apply the algorithm to find the “best” partitioning point
     2. Do this for all possible shifts
     3. Cost $O(m^2)$
  2. More than two clusters
     1. $M$-way partitioning
     2. Try 1, 2, … m-1 split points along the diagonal and try to find the best point for each of these
     3. 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: $A = \bigcup A_{R_i}$
- Reconstruction: $R = \bigotimes_K R_i$, $\forall R_i \in F_R$
- Disjointness:
  - TIDs are not considered to be overlapping since they are maintained by the system
  - Duplicated keys are not considered to be overlapping

Hybrid Fragmentation

![Diagram of Hybrid Fragmentation]
Fragment Allocation

- **Problem:**
  - Given
    - \{F_1, F_2, \ldots, F_n\} Fragments
    - \{S_1, S_2, \ldots, S_m\} Sites
    - \{Q_1, Q_2, \ldots, Q_q\} Applications
  - Find the “optimal” distribution of F to S.

- **Optimality**
  - Minimal cost
    - Communication + Storage + processing
    - Cost is usually in terms of time
  - Performance
    - Response time and/or throughput
  - Constraints
    - Per site constraints (storage and processing)

Information Requirements

- **Database information**
  - Selectivity of fragments
  - Size of fragments
- **Application information**
  - Access types and numbers
  - Access localities
- **Communication 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 database is more complicated
    - Remote file access model is 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 (update) accesses of a query to a fragment
  - A matrix of which queries update which fragments
  - A similar matrix for retrievals
  - Originating site of each query
- Site information
  - Unit cost of storing (processing) data
- 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**
  \[
  x_{ij} = \begin{cases} 
  1 & \text{if Fragment } F_i \text{ is stored at Site } S_j \\
  0 & \text{otherwise}
  \end{cases}
  \]

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

- **Storage Cost** (of Fragment $F_j$ at site $S_k$)
  \((\text{unit cost of storage at } S_k) \times \text{size of } F_j \times x_{jk}\)

- **Query Processing Cost** (for one query)
  \((\text{processing component}) + (\text{transmission component})\)
Allocation Model

• Query Processing Cost
  – Processing component:
    access cost + integrity enforcement cost + concurrency control cost
  – Access cost:
    \[ \sum_{\text{all sites}} \sum_{\text{all fragments}} \left( \text{no. of update accesses} + \text{no. of read accesses} \right) \times x_{ij} \]
    *local processing cost at a site
  – Other costs can be similarly calculated.

Allocation Model

• Query Processing Cost
  – Transmission component:
    cost of processing updates + cost of processing retrievals
  – Cost of updates:
    \[ \sum_{\text{all sites}} \sum_{\text{all fragments}} \text{(update message cost)} + \sum_{\text{all sites}} \sum_{\text{all fragments}} \text{(acknowledgement cost)} \]
  – Retrieval cost:
    \[ \sum_{\text{all fragments}} \min_{\text{all sites}} \left( \text{cost of retrieval command} + \text{cost of sending back the result} \right) \]
Allocation Model

• Constraints:
  – Response Time: execution time of query <= max allowable response time for that query
  – Storage constraint (for a site):
    \[ \sum_{\text{all fragments}} (\text{storage requirements of a fragment at that site}) \leq \text{storage capacity at site} \]
  – Processing constraint (for a site):
    \[ \sum_{\text{all queries}} (\text{processing load of a query at that site}) \leq \text{processing capacity at site} \]

Allocation Model

• Solution Methods:
  – FAP is NP-Complete
  – DAP also NP-Complete
• Heuristic based upon
  – 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” one
  – Ignore replication at first
  – Sliding window on fragments.