Distributed Database Design (Chapter 5)

- **Top-Down Approach:** The database system is being designed from scratch.
  - **Issues:** fragmentation & allocation

- **Bottom-up Approach:** Integration of existing databases (Chapter 15)
  - **Issues:** Design of the export and global schemas.

### Design Consideration (1)

The organization of distributed systems can be investigated along three dimensions:

**Level of sharing**

1. **No sharing:** Each application and its data execute at one site; no communication with other program or access to any data file at other sites
2. **Data sharing:** Programs are replicated at all sites, but data files are not
   - Data is moved to where the query is originated
3. **Data + Program Sharing:** Both data and programs may be shared.
   - A program at one site can request for a service from another program at another site
Design Consideration (2)

Assess Pattern
1. **Static**: Access patterns do not change.
2. **Dynamic**: Access patterns change over time.
   How dynamic the access pattern is?

Level of Knowledge
1. **No information**: Designers do not have the knowledge of the access pattern at all.
2. **Partial information**: Access patterns may deviate from the predictions.
3. **Complete information**: Access patterns can reasonably be predicted and are not much different from the predictions.

---

**TOP-DOWN DESIGN PROCESS**

- **Requirements Analysis**
  - Entity analysis + functional analysis
  - **Conceptual design**
  - **System Requirements (Objectives)**
  - **View Design**
  - **View integration**
  - **Global Conceptual schema**
  - **Access Pattern information**
  - **External Schemas**
  - **Distribution Design**
  - **Maps the local conceptual schemas to the physical storage devices.**
  - **Local Conceptual Schemas**
  - **Physical Design**
  - **Physical Schema**
  - Feedback: Observation and monitoring

- Defining the interfaces for end users
- Involve fragmentation & allocation
Requirement Analysis

- Environment of the system
- Performance, reliability, availability, expandability, and cost.

View Design

Deal with defining the end user interfaces.

Conceptual Design

Consist of

- **Entity Analysis**
  - Determine entities and relationships among them.

- **Functional Analysis**
  - Conceptual design can be seen as an integration of the user views.
  - View integration is used to ensure that the conceptual model support both existing and future applications.
Functional Analysis

- Process of understanding and documenting basic business activities with which the organization is concerned.
- Function is triggered by events and can be defined as tasks that must be carried out as a direct result of an event.
- Output contains
  - Frequency of use of the function
  - How each attributed is used in the function.
  - Requirement on response time, availability, how up-to-date data should be.

Design Issues

- Why fragment at all?
- How should and how much should we fragment?
- A way to test correctness of the fragmentation?
- How to allocate fragments?
- Necessary information for fragmentation and allocation?
Why fragment at all?

- **Performance Reasons:**
  - Interquery concurrency
  - Several queries can be executed in parallel.
  - Intraquery concurrency
  - Allowing parallel execution of a single query.

- **Disadvantages:**
  - Vertical fragmentation may incur overhead.
  - Attributes participating in a dependency may be allocated to different sites.
    - Integrity checking is more costly.
  - If there exists non disjoint fragments → more overhead

---

**Fragmentation Alternatives**

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumental</td>
<td>150,000</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>Database Dev.</td>
<td>135,000</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>250,000</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>Maintenance</td>
<td>350,000</td>
<td>Orlando</td>
</tr>
</tbody>
</table>

**Vertical Partitioning**

<table>
<thead>
<tr>
<th>JNO</th>
<th>BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>150,000</td>
</tr>
<tr>
<td>J2</td>
<td>135,000</td>
</tr>
<tr>
<td>J3</td>
<td>250,000</td>
</tr>
<tr>
<td>J4</td>
<td>310,000</td>
</tr>
</tbody>
</table>

**Horizontal Partitioning**

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumental</td>
<td>150,000</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>Database Dev.</td>
<td>135,000</td>
<td>New York</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumentation</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>Database Devl</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>Maintenance</td>
<td>Orlando</td>
</tr>
</tbody>
</table>
Degree of Fragmentation

• Application views are usually subsets of relations. Hence, it is only natural to consider subsets of relations as distribution units.
  • No fragment → every tuple is a fragment
  • No fragment → every attribute is a fragment

• The appropriate degree of fragmentation is dependent on the applications.

Correctness Rules

• Vertical Partitioning
  • Lossless decomposition
  • Dependency preservation
  • Disjointness on the nonprimary key attributes

• Horizontal Partitioning
  • Disjoint fragments

Allocation Alternatives

• Partitioning: No replication
• Partial Replication: Some fragments are replicated.
• Full Replication: Database exists in its entirety at each site.
Information Requirements for Horizontal Fragmentation

- Database Information
  - Global conceptual schema
  - The relationship between relations
- Application Information

Database Information Notations

\[ \rightarrow 1\text{-to-many relationship} \]

Owner(L1) = S = Source relation
Member(L1) = E = Target relation
Direct Link means equi-join
Application Information

• Qualitative Information
  - The fundamental qualitative information consists of the predicates used in user queries.
  - Analyze user queries based on 80/20 rule: 20% of user queries account for 80% of the total data access.
    ✔️ One should investigate the more important queries.

• Quantitative Information
  - Minterm Cardinality card(m_i): number of tuples that would be accessed by a query specified according to a given minterm predicate.
  - Access Frequency acc(m_i): the access frequency of a given minterm predicate in a given period.

Qualitative information guides the fragmentation activity.
Quantitative information guides the allocation activity.

Simple Predicates

Given a relation R(A_1, A_2, ..., A_n) where A_i has domain D_i, a simple predicate p_j defined on R has the form

\[ p_j: A_i \theta Value \]

where \( \theta \in \{=,\lt,\leq,\gt,\geq\} \) and Value \( \in D_i \)

Example:

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumental</td>
<td>150,000</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>Database Dev</td>
<td>135,000</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>250,000</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>Maintenance</td>
<td>350,000</td>
<td>Orlando</td>
</tr>
</tbody>
</table>

Simple predicates:

\[ p_1: \text{JNAME}= \text{"Maintenance"} \]
\[ p_2: \text{BUDGET} \leq 200,000 \]
Given a set of simple predicates for relation \( R \),
\[
P = \{p_1, p_2, ..., p_m\}
\]
The set of minterm predicates
\[
M = \{m_1, m_2, ..., m_n\}
\]
is defined as
\[
M = \{m_i \mid m_i = \bigwedge_{P \in P} P_i^* \}
\]
where \( P_j^* = p_j \) or \( P_j^* = \neg p_j \).

Possible simple predicates:
- \( P_1 \): TITLE="Elect. Eng."
- \( P_2 \): TITLE="Syst. Analy"
- \( P_3 \): TITLE="Mech. Eng."
- \( P_4 \): TITLE="Programmer"
- \( P_5 \): SAL \leq 35,000
- \( P_6 \): SAL > 35,000

Some corresponding minterm predicates:
- \( m : \) TITLE = "Elect. Eng." \& SAL \leq 35,000
- \( m : \) TITLE \# "Elect. Eng." \& SAL > 35,000

<table>
<thead>
<tr>
<th>TITLE</th>
<th>SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elect. Eng</td>
<td>40,000</td>
</tr>
<tr>
<td>Syst. Analy</td>
<td>54,000</td>
</tr>
<tr>
<td>Mech. Eng</td>
<td>32,000</td>
</tr>
<tr>
<td>Programmer</td>
<td>42,000</td>
</tr>
</tbody>
</table>

Horizontal Fragmentation
- Primary Horizontal Fragmentation
- Derived Horizontal Fragmentation
Primary Horizontal Fragmentation

A primary horizontal fragmentation of a relation is defined by a selection operation on the attribute of the relation.

A possible fragmentation of J is defined as follows:

\[ JP1 = \sigma_{\text{BUDGET} \leq 200,000}(J) \]
\[ JP2 = \sigma_{\text{BUDGET} > 200,000}(J) \]

Horizontal Fragments

Thus, a horizontal fragment \( R_i \) of relation \( R \) consists of all the tuples of \( R \) that satisfy a minterm predicate \( m_i \).

There are as many horizontal fragments (also called minterm fragments) as there are minterm predicates.

Which minterm predicates should we use?

We have to decide on the set of simple predicates that are the basis for the minterm predicates.
Desirable properties of the set of simple predicates

The set should be complete and minimal.

Informally, the set should include only predicates with attributes and conditions that are used in the applications.

Completeness (1)

A set of simple predicate $Pr$ is said to be complete if and only if there is an equal probability of access by every application to any tuple belonging to any minterm fragment that is defined according to $Pr$.

$JP_1 = \sigma_{\text{LOC} = \text{Montreal}}(J)$
$JP_2 = \sigma_{\text{LOC} = \text{New York}}(J)$
$JP_3 = \sigma_{\text{LOC} = \text{Orlando}}(J)$

Case 1: The only application that accesses $J$ wants to access the tuples according to the location (any location). The only possible locations are Orlando, Montreal, and NY. The set of simple predicates $Pr = \{\text{LOC} = \text{Montreal}, \text{LOC} = \text{New York}, \text{LOC} = \text{Orlando}\}$ is complete because each tuple of each fragment has the same probability of being accessed.
Completeness (2)

Example:

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumental</td>
<td>150,000</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>GUI</td>
<td>135,000</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>250,000</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>Database Dev.</td>
<td>310,000</td>
<td>Orlando</td>
</tr>
</tbody>
</table>

Case 2: There is a second application which accesses only those project tuples where the budget is less than $200,000.

The previous Pr is not complete since some tuple in JPi has higher access probability.

- For example, tuple "J2" has higher access probability than tuple "J3" in JP2 (2 applications access J2, but one application accesses J3).
- To make the set complete, we need to add \((BUDGET \leq 200,000, BUDGET > 200,000)\) to Pr.

Example

The only application that accesses J wants to access projects located in New York.

- The set of simple predicates
  
  \[ Pr = \{ LOC = "Montreal", LOC = "New York", Loc = "New York", LOC = "Orlando" \} \]

Is this set Pr complete?

Complete
The only application that accesses J wants to access project located in New York.

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumental</td>
<td>150,000</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>GUI</td>
<td>135,000</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>250,000</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>Database Dev</td>
<td>310,000</td>
<td>Orlando</td>
</tr>
</tbody>
</table>

- Is Loc=“Orlando” relevant? No
- Is Loc=“New York” relevant? Yes
- Is Loc=“Montreal” relevant? No

**Minimality**

**Minimal**

If all the predicates of a set Pr are relevant, Pr is minimal.

That is, if a predicate \( p_i \) causes fragment \( f \) to be fragmented into \( f_i \) and \( f_j \), there should be at least one application that accesses fragments \( f_i \) and \( f_j \) differently.

i.e., The simple predicate \( p_i \) should be relevant in determining a fragmentation.
Relevant

Let \( m_i \) and \( m_j \) be two minterm predicates that are identical in their definition, except that \( m_i \) contains the simple predicate \( p_i \) in its natural-form while \( m_j \) contains \( \neg p_i \).

Also, let \( f_i \) and \( f_j \) be two fragments defined according to \( m_i \) and \( m_j \), respectively. Then \( p_i \) is relevant if and only if

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

\[\text{Access frequency}\]

- \( \text{Card}(fi) \): Number of tuples in \( fi \)

A Complete and Minimal Example

Two applications:

1. One application accesses the tuples according to the location.
2. Another application accesses only those project tuples where the budget is at most $200,000.

Case 1: \( Pr=\{\text{Loc}="Montreal", \text{Loc}="New York", \text{Loc}="Orlando", \ BUDGET<=200,000, BUDGET>200,000\} \) is complete and minimal?

Case 2: If, however, we were to add the predicate \( \text{JNAME}="Instrumentation" \) to \( Pr \), the resulting set would not be minimal since the new predicate is not relevant with respect to the applications.
Case 1: \( Pr = \{ \text{Loc} = \text{Montreal}, \text{Loc} = \text{New York}, \text{Loc} = \text{Orlando}, \text{BUDGET} \leq 200,000, \text{BUDGET} > 200,000 \} \)

\( Pr \) is complete from the previous example. Is \( Pr \) minimal?

- \( J_{11}' \)
- \( J_{12}' \)
- \( J_{1} \)
- \( J_{2} \)

\( \text{Loc} = \text{Orlando} \) is relevant.

---

Case 2

- \( J_{1} \)
- \( J_{2} \)
- \( J_{3} \)
- \( J_{11} \)
- \( J_{12} \)
- \( J_{21} \)
- \( J_{22} \)
- \( J_{31} \)
- \( J_{32} \)

\( \text{LOC} = \text{Montreal} \) is not relevant.
Intuition for “complete”

• Include predicates used in all important applications
• Ensure that you include all the predicates that cover the entire domain of the attribute used in these predicates
• Any two tuples in the same fragment should be accessed by each application in the same way

Example

• The only application that wants to access J tuples with budget greater than 500,000. (Domain of the budget can be any number)

• Is Pr={budget < 200000, budget >= 200000} complete?
Intuition for "Minimal"

- Include only predicates \( p_i \) that are relevant to the applications

- \( p_i \) is relevant if there exists at least one application that
  - uses exactly \( p_i \) or not \( p_i \)
  - and access frequency of \( p_i \) (based on this application) is different from that of not \( p_i \)

Determine the set of meaningful minterm predicates

**Application**: Take the salary and determine a raise accordingly.

**Fragmentation**: The employee records are managed in two places, one handling the records of those with salary less than or equal to $30,000 and the other handling the records of those who earn more than $30,000.

\[ \text{Pr=\{SAL\leq30,000, SAL>30,000\}} \] is complete and minimal.

The minterm predicates:

\[
\begin{align*}
    m: & (SAL \leq 30,000) \land (SAL > 30,000) \\
    m: & (SAL \leq 30,000) \land \neg(SAL > 30,000) \\
    m: & \neg(SAL \leq 30,000) \land (SAL > 30,000) \\
    m: & \neg(SAL \leq 30,000) \land \neg(SAL > 30,000)
\end{align*}
\]

Implications:

\[
\begin{align*}
    i: & (SAL \leq 30,000) \Rightarrow \neg(SAL > 30,000) \\
    i: & \neg(SAL \leq 30,000) \Rightarrow (SAL > 30,000) \\
    i: & (SAL > 30,000) \Rightarrow \neg(SAL \leq 30,000) \\
    i: & \neg(SAL > 30,000) \Rightarrow (SAL \leq 30,000)
\end{align*}
\]

\[ i \Rightarrow m_i \text{ is contradictory} \]
\[ i \Rightarrow m_i \text{ is contradictory} \]

Therefore, we are left with \( M = \{m_2, m_3\} \).
### Invalid Implications

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumental</td>
<td>150,000</td>
<td>Montreal</td>
</tr>
<tr>
<td>J2</td>
<td>Database Dev.</td>
<td>135,000</td>
<td>New York</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>250,000</td>
<td>New York</td>
</tr>
<tr>
<td>J4</td>
<td>Maintenance</td>
<td>350,000</td>
<td>Orlando</td>
</tr>
</tbody>
</table>

#### Simple predicates

- $p_1$: LOC = "Montreal"
- $p_2$: LOC = "New York"
- $p_3$: LOC = "Orlando"
- $p_4$: BUDGET ≤ 200,000
- $p_5$: BUDGET > 200,000

#### VALID Implications

- $i_1: p_1 \Rightarrow \neg p_1 \land \neg p_3$
- $i_2: p_2 \Rightarrow \neg p_1 \land \neg p_3$
- $i_3: p_1 \Rightarrow \neg p_2 \land \neg p_3$
- $i_4: p_1 \Rightarrow \neg p_2$
- $i_5: p_1 \Rightarrow p_3$

#### INVALID Implications

- $i_6: \neg LOC = "Montreal" \Rightarrow \neg (BUDGET > 200,000)$
- $i_7: \neg LOC = "Orlando" \Rightarrow \neg (BUDGET ≤ 200,000)$

Implications should be defined according to the semantics of the database, not according to the current values.

---

### Algorithms

**Rule 1:** fundamental rule of completeness and minimality, which states that a relation or fragment is partitioned "into at least two parts which are accessed differently by at least one application.

- $f_i$ of $Pr'$: fragment $f_i$ defined according to a minterm predicate defined over the simple predicates of $Pr'$.

#### Algorithm 5.1 \texttt{COM.MIN}

**Input:** $R$: relation; $Pr$: set of simple predicates

**Output:** $Pr'$: set of simple predicates

**Declare** $F$: set of minterm fragments

**Begin**

- find a $p_i \in Pr$ such that $p_i$ partitions $R$ according to Rule 1
- $Pr' = p_i$
- $Pr' = Pr - p_i$
- \{ $f_i$ in the minterm fragment according to $p_i$ \}

**Do**

- find a $p_j \in Pr$ such that $p_j$ partitions some $f_i$ of $Pr'$ according to Rule 1
- $Pr'' = p_j$
- $Pr'' = Pr' - p_j$
- $F = F \cup f_i$
- if $F \subseteq Pr'$ which is inadmissable then
  - begin
  - $Pr'' = Pr' - p_j$
  - $F = F - f_i$
  - end-if

**End-Do**

**Until** $Pr'$ is complete

**End.** \texttt{COM.MIN}

#### Algorithm 5.2 \texttt{PHORIZONTAL}

**Input:** $R$: relation; $Pr$: set of simple predicates

**Output:** $M$: set of minterm fragments

**Begin**

- $Pr' = \text{COM.MIN}(R, Pr')$
- determine the set $M$ of minterm predicates
- determine the set $F$ of implications among $p_i \in Pr'$

**For** each $m_i \in M$

- if $m_i$ is contradictory according to $f$ then
  - $M = M - m_i$

**End-For**

**End.** \texttt{PHORIZONTAL}

**Example 5.1**
**Derived Horizontal Fragmentation**

Derived fragmentation is used to facilitate the join between fragments.

In some cases, the horizontal fragmentation of a relation cannot be based on a property of its own attributes, but is derived from the horizontal fragmentation of another relation.

**Benefits of Derived Fragmentation**

**Primary Fragmentation:**

\[
\begin{align*}
PAY_1 &= \sigma_{\text{title} \leq \text{sal}}(PAY) \\
PAY_2 &= \sigma_{\text{title} = \text{sal}}(PAY)
\end{align*}
\]

**Using Derived Fragmentation:**

Simple Join Graph between EMP and PAY

\[
\begin{align*}
\text{EMP}_1 &\rightarrow \text{PAY}_1 \\
\text{EMP}_2 &\rightarrow \text{PAY}_2
\end{align*}
\]

**Simple join graph:** graph with only one link coming in or going out of a fragment.

\[
\begin{align*}
\text{EMP}_i &\rightarrow \text{PAY}_i \text{ can be allocated to the same site.} \\
&\text{Each site can perform a local join independently.}
\end{align*}
\]
Not using derived fragmentation: one can divide EMP into EMP1 and EMP2 based on TITLE and divide PAY into PAY1, PAY2, PAY3 based on SAL. To join EMP and PAY, we have the following scenarios.

More communication overhead!

EMP (ENO, ENAME, TITLE)
PAY (TITLE, SAL)

EMP1
PAY1
EMP2
PAY2
PAY3
PAY3

Derived Fragmentation

EMP (ENO, ENAME, TITLE)  PROJ (PNO, PNAME, BUDGET)

EMP_PROJ (ENO, PNO, RESP, DUR)

• How do we fragment EMP_PROJ?
  - Semi-Join with EMP, or
  - Semi-Join with PROJ

• Criterion: Support the more frequent join operation.
**Star Relationships**

- Design the primary horizontal fragmentation for SPJ.
- Derive the derived fragmentation designs for S, P, and J accordingly.
  - \( S_i = S \cup S_{JNUM} \cup SPJ_i \)
  - \( P_i = P \cup S_{JNUM} \cup SPJ_i \)
  - \( J_i = J \cup S_{JNUM} \cup SPJ_i \)

Exception Case: Primary fragmentation on member relation.

How does the join graph looks like when joining all the relations?

**Chain Relationships**

- Design the primary fragmentation for R1 into M fragments.
- Derive the derived fragmentation for Rk as follows:
  - \( R_k = R_k \cup S_{JNUM} \cup R_{(k-1)PK} \cup R(k-1)i \)
  for \( 2 \leq k \leq N \) in that order for all \( 1 \leq i \leq M \).
Correctness Rules for Horizontal Fragmentation

• Lossless decomposition: If a relation \( R \) is decomposed into fragments \( R_1, \ldots, R_n \), each tuple that can be found in \( R \) can also be found in one or more \( R_i \)’s.

Correctness Rules for Horizontal Fragmentation

• Reconstruction: If relation \( R \) is decomposed into fragments \( R_1, \ldots, R_n \), it should be possible to define a relation operator \( * \) such that

\[
R = *R_i, \text{ for all } R_i
\]

For horizontal fragmentation, \( * \) is the U operator.
• Disjointness: If relation R is horizontally decomposed into fragments R1, R2, ..., Rn and data item di is in Rj, it is not in any other fragment Rk where k!=j.
  - Primary Horizontal Fragmentation
    • Disjointness is guaranteed iff the fragmentation is based on the complete and minimal predicate set
  - Derived Horizontal Fragmentation:
    • Simple join graph
    • Otherwise, investigate actual tuple values

661: Assignment I

ï Critique one of the papers in ACM Multimedia Systems Journal, Volume 10, number 1
ï Available online through our e-journal library
ï Due on Sept 16, 2004
ï % contributed to the final grade: 7%
ï Total score: 100 points
ï The critique includes
  ñ Your name and last four digits of your university ID
  ñ (20 points) Discuss originality of the work (what is new in this paper that has never been done previously)
  ñ (50 points) 2 page summary of the proposed work (how does the technique work, how performance evaluation is performed)
  ñ (20 points) Discuss drawbacks of the proposed work (in your opinion)
    ï Drawback of the proposed technique
    ï Drawback of the performance evaluation
  ñ (10 points) What would you do differently to solve the same problem?
### September 04

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
<th>Sunday</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>August 30</td>
<td>31</td>
<td>September 1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Assignment 1: plan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assignment 1: Due</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>October 1</td>
<td>2</td>
</tr>
<tr>
<td>Assignment 2: Exercise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### October 04

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
<th>Sunday</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>October 2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>Grade book</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assignment 2: Due</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>
**Vertical Fragmentation Approaches**

**Greedy Heuristic Approaches:**

- **Split Approach:** Global relations are progressively split into fragments.

- **Grouping Approach:** Attributes are progressively aggregated to constitute fragments.

**Correctness:**

- **Lossless decomposition:** Each attribute of $R$ belongs to at least one fragment; each fragment includes either a key of $R$ or a "tuple identifier".
- **Reconstruction (using join operators)**
- **Disjointness for non-key attributes (not part of primary keys)**
Vertical Clustering—Replication without Fragments

In evaluating the convenience of vertical clustering, it is important that overlapping attributes are not heavily updated.

Example:

Bad Fragmentation: NAME not available in EMP2
1. EMP1(ENUM, NAME, TAX, SAL)
2. EMP2(ENUM, MGRNUM, DNUM)

Good Fragmentation: NAME is relatively stable.
1. EMP1(ENUM, NAME, TAX, SAL)
2. EMP2(ENUM, NAME, MGRNUM, DNUM)

Split Approach

• Splitting is considered only for attributes that do not participate in the primary key.

• The split approach involves three steps:

  1. Obtain attribute affinity matrix: this matrix tells how closely related attributes are.

  2. Use a clustering algorithm to group some attributes together based on the attribute affinity matrix. This algorithm produces a clustered affinity matrix.

  3. Use a partitioning algorithm to partition attributes such that sets of attributes are accessed solely or for the most part by distinct set of applications.
Step 1: Obtain attribute affinity matrix (1)

<table>
<thead>
<tr>
<th>PROJ</th>
<th>PNO</th>
<th>PNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>A4</td>
<td></td>
</tr>
</tbody>
</table>

q1: SELECT BUDGET FROM PROJ WHERE PNO=Value;

q2: SELECT PNAME, BUDGET FROM PROJ;

q3: SELECT PNAME FROM PROJ WHERE LOC=Value;

q4: SELECT SUM(BUDGET) FROM PROJ WHERE LOC=Value

Attribute Usage Matrix

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>q2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>q3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>q4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

One application corresponds to one query.

Attribute Affinity Matrix (2)

We need another matrix to include application frequency. This metric is called attribute affinity measure.

\[
\text{aff}(A_i, A_j) = \sum_{k=1}^{n} \sum_{l=1}^{m} \text{ref}_i(q_k) \cdot \text{acc}_l(q_k)
\]

\[
\text{aff}(A_i, A_j): \text{# times the two attributes are accessed together, taking into considerations of all sites}
\]

\[
\text{RefI}(q_k): \text{Number of accesses to attributes (A}_i, \text{A}_j) \text{ for each execution of q}_k \text{ at site l}
\]

\[
\text{Accl}(q_k): \text{Application access frequency of q}_k \text{ at site l}
\]
\[ \text{aff}(A_i, A_j) = \sum_{k \mid \text{use}(q_k, A_j) = 1} \sum_{l \mid \text{use}(q_k, A_l) = 1} \text{ref}_i(q_k) \text{acc}_j(q_k) \]

**Example:**
Let \( \text{ref}_i(q_k) = 1 \) for all \( q_k \) at site \( i \). The application frequencies are

<table>
<thead>
<tr>
<th>Site 1</th>
<th>Site 2</th>
<th>Site 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{acc}_1(q_1) = 15 )</td>
<td>( \text{acc}_2(q_2) = 20 )</td>
<td>( \text{acc}_3(q_3) = 10 )</td>
</tr>
<tr>
<td>( \text{acc}_1(q_2) = 5 )</td>
<td>( \text{acc}_2(q_2) = 0 )</td>
<td>( \text{acc}_3(q_3) = 0 )</td>
</tr>
<tr>
<td>( \text{acc}_1(q_3) = 25 )</td>
<td>( \text{acc}_2(q_3) = 25 )</td>
<td>( \text{acc}_3(q_3) = 25 )</td>
</tr>
<tr>
<td>( \text{acc}_1(q_4) = 3 )</td>
<td>( \text{acc}_2(q_4) = 0 )</td>
<td>( \text{acc}_3(q_4) = 0 )</td>
</tr>
</tbody>
</table>

\( \text{aff}(A_1, A_3) = \text{acc}_1(q_1) + \text{acc}_2(q_1) + \text{acc}_3(q_1) = 45 \) since \( q_1 \) is the only application that accesses both attributes.

**Attribute Affinity Matrix (2)**

\[
\begin{array}{cccc}
A_1 & A_2 & A_3 & A_4 \\
q_1 & 1 & 0 & 1 & 0 \\
q_2 & 0 & 1 & 1 & 0 \\
q_3 & 0 & 1 & 0 & 1 \\
q_4 & 0 & 0 & 1 & 1 \\
\end{array}
\quad
\begin{array}{cccc}
A_1 & A_2 & A_3 & A_4 \\
A_1 & 45 & 0 & 45 & 0 \\
A_2 & 0 & 80 & 5 & 75 \\
A_3 & 45 & 5 & 53 & 3 \\
A_4 & 0 & 75 & 3 & 78 \\
\end{array}
\]

**Example:**
Let \( \text{ref}_i(q_k) = 1 \) for all \( q_k \) at site \( i \). The application frequencies are

<table>
<thead>
<tr>
<th>Site 1</th>
<th>Site 2</th>
<th>Site 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{acc}_1(q_1) = 15 )</td>
<td>( \text{acc}_2(q_2) = 20 )</td>
<td>( \text{acc}_3(q_3) = 10 )</td>
</tr>
<tr>
<td>( \text{acc}_1(q_2) = 5 )</td>
<td>( \text{acc}_2(q_2) = 0 )</td>
<td>( \text{acc}_3(q_3) = 0 )</td>
</tr>
<tr>
<td>( \text{acc}_1(q_3) = 25 )</td>
<td>( \text{acc}_2(q_3) = 25 )</td>
<td>( \text{acc}_3(q_3) = 25 )</td>
</tr>
<tr>
<td>( \text{acc}_1(q_4) = 3 )</td>
<td>( \text{acc}_2(q_4) = 0 )</td>
<td>( \text{acc}_3(q_4) = 0 )</td>
</tr>
</tbody>
</table>
• Cluster attributes together based on the BEA by permutes rows and columns to maximize the global affinity measure

• Computation time: $O(n^2)$ where $n$ is the number of attributes

---

**Generating Clustered Affinity Matrix** [1984] using Bond Energy Algorithm (BEA) [1972]

**Step 1: Initialize CA**

![Diagram showing the initialization process of the Clustered Affinity Matrix (CA)]
\textbf{Step 2: Determine the order for }A_3\textbf{ }

\begin{align*}
\text{Cont}(A_0,A_1,A_3) &= 8820 \\
\text{Cont}(A_1,A_3,A_2) &= 10150 \\
\text{Cont}(A_2,A_3,A_4) &= 1780 \\
\end{align*}

(A3 is left most column) \hspace{1cm} (A3 is between A1 and A2) \hspace{1cm} (A3 is between A2 and A4)

Since Cont\((A_1,A_3,A_2)\) is the biggest, \([A_1,A_3,A_2]\) is the best order.

\begin{align*}
\text{Cont}(A_0,A_3,A_1) &= 2(\text{bond}(A_0,A_3) + \text{bond}(A_3,A_1) - \text{bond}(A_1,A_3)) \\
\text{bond}(A_0,A_3) &= \sum_{z=1}^{n} \text{aff}(A_0,A_z) \cdot \text{aff}(A_z,A_3) \\
\end{align*}

\begin{align*}
\text{Cont}(A_0,A_3,A_1) &= 2(45 \cdot 45 + 5 \cdot 0 + 53 \cdot 45 + 3 \cdot 0) = 8820 \\
\text{bond}(A_0,A_3) &= 45 \cdot 45 + 5 \cdot 0 + 53 \cdot 45 + 3 \cdot 0 = 8820 \\
\text{Cont}(A_1,A_3,A_2) &= 2(45 \cdot 45 + 5 \cdot 80 + 53 \cdot 45 + 3 \cdot 0) = 10150 \\
\text{bond}(A_1,A_3) &= 45 \cdot 45 + 5 \cdot 80 + 53 \cdot 45 + 3 \cdot 0 = 10150 \\
\text{Cont}(A_2,A_3,A_4) &= 2(45 \cdot 45 + 5 \cdot 0 + 53 \cdot 45 + 3 \cdot 0) = 1780 \\
\text{bond}(A_2,A_3) &= 45 \cdot 45 + 5 \cdot 0 + 53 \cdot 45 + 3 \cdot 0 = 1780
\end{align*}

\textbf{Boundary Condition:} \text{aff}(A_0,A_j) = \text{aff}(A_i,A_0) = \text{aff}(A_{n+1},A_j) = \text{aff}(A_i,A_{n+1}) = 0
Step 2: Determine the order for $A_4$

Since $\text{Cont}(A_3,A_2,A_4)$ is the biggest, $[A_3,A_2,A_4]$ is the best order.

Clustered Affinity Matrix (CA)

Step 3: Re-order the Rows

The rows are organized in the same order as the columns.

Clustered Affinity Matrix (CA) from Step 2.
Bond Energy Clustering Algorithm

Algorithm 5.3 BEA

input: AA: attribute affinity matrix
output: CA: clustered affinity matrix

begin
(initialize; remember that AA is an $n \times n$ matrix)

$CA(i, 1) \leftarrow AA(i, 1)$
$CA(i, 2) \leftarrow AA(i, 2)$

index $\leftarrow 3$

while index $\leq n$ do (choose the “best” location for attribute $AA_{index}$)

begin

for $i$ from 1 to index - 1 by 1 do

calculate $cont(A_{i-1}, A_{index}, A_i)$

end-for

calculate $cont(A_{index-1}, A_{index}, A_{index+1})$ (boundary condition)

loc $\leftarrow$ placement given by maximum $cont$ value

for $j$ from index to loc by -1 do

(reshape the two matrices)

$CA(i, j) \leftarrow CA(i, j - 1)$

end-for

$CA(i, loc) \leftarrow AA(i, index)$

index $\leftarrow index + 1$

end-while

order the rows according to the relative ordering of columns

end (BEA)

Excerpt from [Ozsu 1999]

---

Partitioning

Find the sets of attributes that are accessed, for the most part, by distinct sets of applications.

We look for dividing points along the diagonal such that

- total accesses to only one fragment are maximized, while
- total accesses to more than one fragments are minimized.

Clustered Affinity Matrix (CA)

Cluster 1: $A_1$ & $A_3$
Cluster 2: $A_2$ & $A_4$

Two vertical fragments: PROJ1($A_1, A_3$) and PROJ2($A_2, A_4$)

Need to replicate key attribute for PROJ$i$
Applications are classified into three sets:

- **TQ**: Applications that only use attributes in TA.
- **BQ**: Applications that only use attributes in BA.
- **OQ**: Applications that access both TQ and BQ.

2-Way Partitioning

Find the partitioning point \( x \) (1 \( \leq x \leq n \)) such that cost \( Z \) is maximized.

\[
Z = CTQ \cdot CBQ - COQ^2
\]

Implication: CTQ is about CBQ for load balancing purposes.

Complexity: \( O(n) \)

CTQ = \( \sum_{q \in TQ} \sum_{j} ref_j(q) \cdot acc_j(q) \)

- **CTQ**: Total number of accesses to attributes by TQ.
- **CBQ**: Total number of accesses to attributes by BQ.
- **COQ**: Total number of accesses to attributes by OQ.

HYBRID FRAGMENTATION

- Apply horizontal fragmentation to vertical fragments.
- Apply vertical fragmentation to horizontal fragments.

Example: Applications about work at each department reference tuples of employees in the departments located around the site with 80% probability.

- **EMP(ENUM, NAME, SAL, TAX, MGRNUM, DNUM)**
  - **Jacksonville**
  - **Orlando**
  - **Miami**

Vertical fragmentation

Horizontal Fragmentation