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

- **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

#### 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>
<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>

#### JP1

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Instrumentation</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>

#### JP2

<table>
<thead>
<tr>
<th>JNO</th>
<th>JNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<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>310,000</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**

- **S**: Title, SAL
- **E**: ENO, ENAME, TITLE
- **J**: JNO, JNAME, BUDGET, LOC
- **G**: ENO, JNO, RESP, DUR

→ 1-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 Selectivity sel(mᵢ): number of tuples that would be accessed by a query specified according to a given minterm predicate.
  - Access Frequency acc(mᵢ): 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(A₁, A₂, ..., Aₙ) where Aᵢ has domain Dᵢ, a simple predicate pᵢ defined on R has the form

\[ pᵢ: Aᵢ \theta Value \]

where \( \theta \in \{=,\!<\!,\!<\!,\!<\!,\!\geq\!\} \) and Value \( \in Dᵢ \)

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:

\[ p₁: \text{JNAME} = \text{"Maintenance"} \]
\[ p₂: \text{BUDGET} \leq 200,000 \]
Given a set of simple predicates for relation $R$,
$P = \{p_1, p_2, \ldots, p_m\}$
The set of minterm predicates
$M = \{m_1, m_2, \ldots, m_n\}$
is defined as
$M = \{m_i \mid m_i = \bigwedge_{p \in P} P_j^* \}$
where $P_j^* = p_j$ or $P_j^* = \neg p_j$.

Minterm predicate is a conjunction of simple predicates

### Possible simple predicates:
- $P_1$: TITLE="Elect. Eng."
- $P_2$: TITLE="Syst. Analy"
- $P_4$: TITLE="Programmer"
- $P_5$: SAL<=35,000
- $P_6$: SAL > 35,000

### Some corresponding minterm predicates:
- $m_1$: TITLE="#Elect.Eng.#\land SAL \leq 35,000$
- $m_2$: TITLE="#Elect.Eng.#\land SAL > 35,000$
Primary Horizontal Fragmentation

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

\[
\begin{align*}
E & \quad \text{ENO} \quad \text{ENAME} \quad \text{TITLE} \\
& \quad \text{JNO} \quad \text{JNAME} \quad \text{BUDGET} \quad \text{LOC} \\
& \quad \text{G} \quad \text{ENO} \quad \text{JNO} \quad \text{RESP} \quad \text{DUR} \\
L_2 & \quad \text{L}_3
\end{align*}
\]

Owner(\text{L}_3) = J, Member(\text{L}_3) = \text{G}

A possible fragmentation of J is defined as follows:

\[
\begin{align*}
\text{JP}_1 &= \sigma_{\text{BUDGET} \leq 200,000}(J) \\
\text{JP}_2 &= \sigma_{\text{BUDGET} > 200,000}(J)
\end{align*}
\]

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 $Pr$.

Case 1: The only application that accesses $J$ wants to access the tuples according to the location (any location).

The set of simple predicates

$$Pr = \begin{cases} LOC = "Montreal" \\ LOC = "New York" \\ LOC = "Orlando" \end{cases}$$

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 access J3).
- To make the set complete, we need to add (BUDGET ≤ 200,000, BUDGET > 200,000) to Pr.

Note: Completeness is a desirable property because a complete set defines fragments that are not only logically uniform in that they all satisfy the minterm predicate, but statistically homogeneous.

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?
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>290,000</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, 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.
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

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

- Card($f_i$): Number of tuples in $f_i$

---

**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=$\{\text{Loc}="Montreal", \text{Loc}="New York", \text{Loc}="Orlando", \text{BUDGET}<=200,000, \text{BUDGET}>200,000\}$ is complete and minimal?

**Case 2**: If, however, we were to add the predicate 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?

- **J1** Instrumental 150,000 Montreal
- **J2** Database Dev. 135,000 New York
- **J3** CAD/CAM 250,000 New York
- **J4** Maintenance 350,000 Orlando

Loc='Orlando' is relevant.

**Case 2**

- **J1** Instrumental 150,000 Montreal
- **J2** Database Dev. 135,000 New York
- **J3** CAD/CAM 250,000 New York
- **J4** Maintenance 350,000 Orlando

[ JNAME = "Instrument" ] is not relevant.
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.

Pr={SAL≤30,000, SAL>30,000} is complete and minimal.

The minterm predicates:

\[
m_i : (SAL \leq 30,000) \land (SAL > 30,000)
\]

Implications:

\[
i_i : (SAL \leq 30,000) \Rightarrow (SAL > 30,000)
i_i : (SAL > 30,000) \Rightarrow (SAL \leq 30,000)
\]

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**

\[
i : p_i \Rightarrow \neg p_i \land \neg p_i
\]

**INVALID Implications**

\[
i : LOC="Montreal" \Rightarrow \neg (BUDGET > 200,000)
i : LOC="Orlando" \Rightarrow \neg (BUDGET \leq 200,000)
\]

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, of Pr: fragment f, defined according to a minterm predicate defined over the simple predicates of Pr.

---

**Algorithm 5.1** 
**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 ∈ Pr such that p partitions R according to Rule 1  
P∗ = p  
P = Pr − p  
F = f  
{f, is the minterm fragment according to p1}  
do  

**begin**  
find a p ∈ Pr such that p partitions some f of Pr’ according to Rule 1  
P∗ = Pr ∩ p  
P = Pr − p  
F = F ∪ f  
if p is a property which is not relevant then  
begin  
P∗ = Pr − p  
F = F − f  
end-if  
end-begin  
until Pr’ is complete  
end. (COM.MIN)

---

**Algorithm 5.2** 
**HORIZONTAL**

**input:** R, relation; Pr: set of simple predicates  
**output:** M: set of minterm fragments  
**begin**  
P∗ = COM.MIN(R, Pr)  
determine the set M of minterm predicates  
determine the set f of implications among p ∈ Pr  
for each m ∈ M do  
if m is contradictory according to f then  
M = M − m  
end-if  
end-for  
end. (HORIZONTAL)

---

**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:

\[
\begin{align*}
PAY_1 &= \sigma_{\text{TITLE}}(PAY) \\
PAY_2 &= \sigma_{\text{SAL}}(PAY)
\end{align*}
\]

Using Derived Fragmentation:

Simple Join Graph between EMP and PAY

EMP1 \rightarrow PAY1 \\
EMP2 \rightarrow PAY2

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.

EMP1 \rightarrow PAY1 \\
EMP1 \rightarrow PAY2 \\
EMP1 \rightarrow PAY3 \\
EMP2 \rightarrow PAY2 \\
EMP2 \rightarrow PAY3

EMP_i and PAY_i can be allocated to the same site.

More communication overhead!

Derived Fragmentation

EMP (ENO, ENAME, TITLE) \rightarrow 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.
  - \( S_i = S_{SNUM, SPJ_i} \)
  - \( P_i = P_{SNUM, SPJ_i} \)
  - \( J_i = J_{SNUM, SPJ_i} \)

**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 \( R_k \) as follows:
  - \( R_k = R_{RFK, R(k-1)FK} \) for \( 2 \leq k \leq N \) in that order for all \( 1 \leq i \leq M \).