Distributed DB Design

• Multi-DBs (or bottom-up)
  - No design issues!
• Top-down approach
  - Given a DB (and workload), how to split and allocate to sites
  - Two design issues
    • Fragmentation
    • Allocation
  - Issues not independent but will cover separately

Example
Employee relation E (#, name, loc, sal, ...)

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>joe</td>
<td>Sa</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Tom</td>
<td>Sa</td>
<td>15</td>
</tr>
</tbody>
</table>

40% of queries:
Qa: select * from E where loc=Sa and ...
Qb: select * from E where loc=Sb and ...

Motivation: Two sites: Sa, Sb
Qa \(\rightarrow\) Sa \(\leftarrow\) Qb

\(\mathcal{F} = \{ F_1, F_2 \} \)

\(F_1 = \sigma_{\text{loc}=\text{Sa}} E \quad F_2 = \sigma_{\text{loc}=\text{Sb}} E\)

\(\Rightarrow\) called primary horizontal fragmentation
\(\Rightarrow\) fragments are expressed in R.A

Distributed DB Design

• It does not take a rocket scientist to figure out fragmentation...

\[ E \]

\[ \mathcal{F} \]

\[ \mathcal{F} = \{ F_1, F_2 \} \]

\(F_1 = \sigma_{\text{loc}=\text{Sa}} E \quad F_2 = \sigma_{\text{loc}=\text{Sb}} E\)

\(\Rightarrow\) called primary horizontal fragmentation
\(\Rightarrow\) fragments are expressed in R.A

Fragmentation

- Horizontal: Primary depends on local attributes
- Vertical: Derived depends on foreign relation
Which are good fragmentations?

Example 1:
\[ F = \{ F_1, F_2 \} \]
\[ F_1 = \sigma_{\text{sal}<10} E \]
\[ F_2 = \sigma_{\text{sal}>20} E \]

Problem: Some tuples lost!

Example 2:
\[ F = \{ F_3, F_4 \} \]
\[ F_3 = \sigma_{\text{sal}<10} E \]
\[ F_4 = \sigma_{\text{sal}>5} E \]

Tuples with 5 < sal < 10 are duplicated...

⇒ Prefer to deal with replication explicitly

Example: \[ F = \{ F_5, F_6, F_7 \} \]
\[ F_5 = \sigma_{\text{sal} \leq 5} E \]
\[ F_6 = \sigma_{5< \text{sal} <10} E \]
\[ F_7 = \sigma_{\text{sal} \geq 10} E \]

← Then replicate \( F_6 \) if convenient
(part of allocation problem)

Desired properties for horizontal fragmentation

\[ R \Rightarrow F = \{ F_1, F_2, \ldots \} \]

(1) Completeness
\[ \forall t \in R, \exists F_i \in F \text{ such that } t \in F_i \]

(2) Disjointness
\[ \forall t \in F_i, \neg \exists F_j \text{ such that } t \in F_j, i \neq j, F_i, F_j \in F \]

(3) Reconstruction
\[ \bigcup F_i \]

How many fragments to generate? How to decide?

Example of generation

- Say queries use predicates: \( A<10 \), \( A>5 \), \( \text{Loc} = \text{Sa} \), \( \text{Loc} = \text{Sb} \)
- Assumption: Only 2 locations
- Next: - generate “minterm” predicates
- eliminate useless ones
Minterm predicates (part I)

1. \(A < 10 \land A > 5 \land \text{Loc} = \text{SA} \land \text{Loc} = \text{SB}\)
2. \(A < 10 \land A > 5 \land \text{Loc} = \text{SA} \land \neg \text{Loc} = \text{SB}\)
3. \(A < 10 \land A > 5 \land \neg \text{Loc} = \text{SA} \land \text{Loc} = \text{SB}\)
4. \(A < 10 \land A > 5 \land \neg \text{Loc} = \text{SA} \land \neg \text{Loc} = \text{SB}\)

Minterm predicates (part II)

5. \(A < 10 \land \neg (A > 5) \land \text{Loc} = \text{SA} \land \text{Loc} = \text{SB}\)
6. \(A < 10 \land \neg (A > 5) \land \neg \text{Loc} = \text{SA} \land \text{Loc} = \text{SB}\)
7. \(A < 10 \land \neg (A > 5) \land \neg \text{Loc} = \text{SA} \land \neg \text{Loc} = \text{SB}\)
8. \(A < 10 \land \neg (A > 5) \land \neg \text{Loc} = \text{SA} \land \neg \text{Loc} = \text{SB}\)

Final fragments:

F2: \(5 < A < 10 \land \text{Loc} = \text{SA}\)
F3: \(5 < A < 10 \land \text{Loc} = \text{SB}\)
F6: \(A \leq 5 \land \text{Loc} = \text{SA}\)
F7: \(A \leq 5 \land \text{Loc} = \text{SB}\)
F10: \(A \geq 10 \land \text{Loc} = \text{SA}\)
F11: \(A \geq 10 \land \text{Loc} = \text{SB}\)

Note: elimination of useless fragments depends on application semantics:

e.g.: if LOC could be \(\neq \text{SA}, \neq \text{SB}\),
we need to add fragments

F4: \(5 < A < 10 \land \text{Loc} \neq \text{SA} \land \text{Loc} \neq \text{SB}\)
F8: \(A \leq 5 \land \text{Loc} \neq \text{SA} \land \text{Loc} \neq \text{SB}\)
F12: \(A \geq 10 \land \text{Loc} \neq \text{SA} \land \text{Loc} \neq \text{SB}\)

Summary

- Given simple predicates \(Pr = \{ p_1, p_2, \ldots, p_m \}\)
  - Minterm predicates are
    \[ M = \{ m \mid m = \bigwedge_{p \in Pr} p^*, \ 1 \leq k \leq m \} \]
  - where \(p^*\) is \(p\) or \(\neg p\)
- Fragments \(\sigma_m R\) for all \(m \in M\) are complete and disjoint

Which simple predicates should we use in \(Pr\)?

\(\Leftrightarrow\) Desired property of \(Pr\):
- minimality
- completeness

\(\text{different from COMPLETENESS of fragmentation!}\)
Return to example:
E(#, NM, LOC, SAL,...)
Common queries:
Qa: select * from E
where LOC=Sa and ...
Qb: select * from E
where LOC=Sb and ...

Three choices:
(1) Pr = {}  F
(2) Pr = {LOC=Sa, LOC=Sb}  F2={σ loc=Sa E, σ loc=Sb E}
(3) Pr = {LOC=Sa, LOC=Sb, Sal<10}  F3={σ loc=Sa ∧ sal<10 E, σ loc=Sa ∧ sal≥10 E, σ loc=Sb ∧ sal<10 E, σ loc=Sb ∧ sal≥10 E}

In other words:

Qa: Select ... loc = Sa ...
Qb: Select ... loc = Sa ...

F2 is good... (not F1, F3)

Informal definition
Set of predicates Pr is complete if for every Fi ∈ F[Pr],
every t ∈ Fi has equal probability of access by every (major) application.

So F2 is “good”...

Back to example:

Qa: Select ... loc = Sa ...
Qb: Select ... loc = Sa ...

So F1 is not “good”...

Another back to example:

Qa: Select ... loc = Sa ...
Qb: Select ... loc = Sa ...

So F2 is “good”...

So is F3...
Is Pr complete a good thing?

Informal definition
Set of predicates Pr is minimal if no Pr \subset Pr is complete

Back to example:

<table>
<thead>
<tr>
<th>1</th>
<th>Pr = {}</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Pr = {LOC=Sa, LOC=Sb}</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>Pr = {LOC=Sa, LOC=Sb, Sal&lt;10}</td>
<td>✓</td>
</tr>
</tbody>
</table>

Pr(2) is a subset of Pr(3), so Pr(3) is not minimal...

How do we get complete and minimal Pr?

Answer: use predicates that are "relevant" in frequent queries

Example: Qa:
Select *
from E
where LOC=Sa and
SAL > input parameter

Derived horizontal fragmentation

Example:
E(#, NM, SAL, LOC)

Fragmentation of E: \mathcal{F} = \{E_1, E_2\} by LOC J(#, DES, ...)

Common query for project:

Given employee name,
list projects (s)he works in

E \bowtie J = (E_1 \cup E_2) \bowtie J
= (E_1 \bowtie J) \cup (E_2 \bowtie J)

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Joe</td>
<td>Sa</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Tom</td>
<td>Sa</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Fred</td>
<td>Sb</td>
<td>15</td>
</tr>
</tbody>
</table>

How to fragment J?
On which attribute?
How to do it?
(by looking at J alone?)
Derived horizontal fragmentation

\[ R, \mathcal{F} = \{ F_1, F_2, \ldots, F_n \} \]
\[ \Downarrow \]
\[ S, \mathcal{D} = \{ D_1, D_2, \ldots, D_n \} \] where \( D_i = S \bowtie F_i \)

Convention: \( R \) is owner
\( S \) is member

Checking completeness and disjointness of derived fragmentation

Example: Say \( J \) is:

<table>
<thead>
<tr>
<th>#</th>
<th>Des</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>33</td>
<td>project 33</td>
</tr>
</tbody>
</table>

But no \# = 33 in \( E_1 \) or in \( E_2 \)!

This \( J \) tuple will not be in \( J_1 \) or \( J_2 \)

Fragmentation not complete

Example:

To get completeness …

- Need to enforce referential integrity constraint:
  
  \[ \text{join attr(#)} \text{ of member relation} \]
  
  \[ \Downarrow (\subseteq) \]
  
  \[ \text{join attr(#)} \text{ of owner relation} \]

Example:

To get disjointness …

- Join attribute(#) should be key of owner relation
Summary: horizontal fragmentation

- Type: primary, derived
- Properties: completeness, disjointness
- Predicates: minimal, complete

Vertical fragmentation (When?)

Example:

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Joe</td>
<td>Sa</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Fred</td>
<td>Sa</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>NM</th>
<th>Loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Joe</td>
<td>Sa</td>
</tr>
<tr>
<td>7</td>
<td>Sally</td>
<td>Sb</td>
</tr>
<tr>
<td>8</td>
<td>Fred</td>
<td>Sa</td>
</tr>
</tbody>
</table>

Properties:

\[ R[T] \Rightarrow R_i[T_i] \quad T_i \subseteq T \]

(1) Completeness

\[ \bigcup_{i} T_i = T \]

(2) Disjointness

\[ T_i \cap T_j = \emptyset \quad \text{for all } i, j \neq j \]

(3) Lossless join

\[ \forall i \quad R_i = R \]

One way to achieve lossless join:
Repeat key in all fragments, i.e.,
Key \subseteq T_i \quad \forall i
How do we decide what attributes are grouped with which?

Example:

\[
E(#,NM,LOC,SAL) = E_1(#,NM) \cup E_2(#,LOC) \cup E_3(#,SAL)
\]

Attribute affinity matrix

\[
\begin{array}{cccccc}
A_1 & A_2 & A_3 & A_4 & A_5 \\
A_1 & - & - & - & - & - \\
A_2 & 50 & - & - & - & - \\
A_3 & 45 & 48 & - & - & - \\
A_4 & 1 & 2 & 0 & - & - \\
A_5 & 0 & 0 & 4 & 75 & - \\
\end{array}
\]

\(R_1[K,A_1,A_2,A_3]\) and \(R_2[K,A_4,A_5]\)

Allocation

Example: \(E(#,NM,LOC,SAL) \Rightarrow F_1 = \sigma_{loc=Sa} E ; F_2 = \sigma_{loc=Sb} E\)

Qa: select ... where loc=Sa...
Qb: select ... where loc=Sb...

Where do \(F_1,F_2\) go?

Site a

Site b

Many Issues ...

• Where do queries originate
• What is communication cost?
  and size of answers, relations,...?
• What is storage capacity, cost at sites?
  and size of fragments?
• What is processing power at sites?
• What is query processing strategy?
  - How are joins done?
  - Where are answers collected?
• Do we replicate fragments?
  - Cost of updating copies?
  - Writes and concurrency control?

Optimization problem

• What is best placement of fragments
  and/or best number of copies to:
  - minimize query response time
  - maximize throughput
  - minimize "some cost"
  - ...
• Subject to constraints?
  - Available storage
  - Available bandwidth, power,...
  - Keep 90% of response time below X
  - ...

This is an incredibly hard problem!
**Example: Single fragment F**

Read cost: \( \sum_{i=1}^{m} [t_i \times \text{MIN } C_{ij}] \)

- **i**: Originating site of request
- **t_i**: Read traffic at S_i
- **C_{ij}**: Retrieval cost

Accessing fragment F at S_j from S_i

**Scenario - Read cost**

![Diagram of read cost]

**Write cost**

\( \sum_{i=1}^{m} \sum_{j=1}^{m} X_{ij} u_i C'_{ij} \)

- **i**: Originating site of request
- **j**: Site being updated
- **X_{ij}**: 0 if F not stored at S_j
  1 if F stored at S_j
- **u_i**: Write traffic at S_i
- **C'_{ij}**: Write cost

Updating F at S_j from S_i

**Scenario - write cost**

![Diagram of write cost]

**Storage cost:**

\( \sum_{i=1}^{m} X_i d_i \)

- **X_i**: 0 if F not stored at S_i
  1 if F stored at S_i
- **d_i**: storage cost at S_i

**Target function**

\[
\min \left\{ \sum_{i=1}^{m} [t_i \times \text{MIN } C_{ij}] + \sum_{j=1}^{m} X_j \times u_i \times C'_{ij} \right\} + \sum_{i=1}^{m} X_i \times d_i \]
Can add more complications:

Examples:
- Multiple fragments
- Fragment sizes
- Concurrency control cost

Example

Assumptions:
- The cost of storing fragments at sites is negligible.
- The cost of reading a non-local fragment is 1 unit and the cost of writing a non-local fragment is 2 units.
- The cost of reading or writing a local fragment is 0 unit.
- app1 reads fragment F1 5 times and reads fragment F2 5 times.
- app2 reads fragment F2 5 times and reads fragment F3 5 times.
- app3 reads fragment F2 10 times and writes fragment F2 10 times.
- app1 issues at site 1, app2 issues at site 2, and app3 issues at site 3.

Which of the following three allocations of fragments minimizes total cost?
- 1: F1 at site 1; F2 at site 2; F3 at site 3
- 2: F1 and F2 at site 1; F2 and F3 at site 3.
- 3: F1 at site 1, F3 at site 2, F2 at site 3.

Example (Cont)

Allocation-1: F1 at site 1; F2 at site 2; F3 at site 3
App1 5 non-local reads cost 5
App2 5 non-local read cost 5
App3 10 non-local reads cost 10, non-local writes cost 20.
Total cost is 40.

Allocation-2: F1 and F2 at site 1; F2 and F3 at site 3.
App1 costs 0
App2 10 non-local reads cost 10
App3 10 non-local writes cost 20
Total cost is 30.

Allocation-3: Total is 10.

“Best-fit” strategy for non-replicated allocation

- Place the fragment i at the site j* where the total costs/savings to i is minimum/maximum
- What would be the allocation result (of our example) if we use the “best-fit” method?
- The number of total references for fragment F2 is 5+5+10+10x2 = 40.
  - When allocating F2 to sites 1, 2, and 3, the total cost will be 35, 35, and 10, respectively.
  - Correspondingly, the total saving will be 5, 5, and 30.
  Under “best-fit”, F2 is placed at site 3.

Summary

- Description of fragmentation
- Good fragmentations
- Design of fragmentation
- Allocation