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, …)

40% of queries:

Qa: select *
from E
where loc=Sa
and...

40% of queries:

Qb: select *
from E
where loc=Sb
and ...

Motivation: Two sites: Sa, Sb

Qa → Sa

Sb ← Qb

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

<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>

At Sa

<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>
</tbody>
</table>

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

\[ F_1 = \sigma_{\text{loc}=\text{Sa}} E \quad F_2 = \sigma_{\text{loc}=\text{Sb}} E \]

\[ \Rightarrow \text{called primary horizontal fragmentation} \]
\[ \Rightarrow \text{fragments are expressed in R.A} \]
Which are good fragmentations?

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

Problem: Some tuples lost!

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

Tuples with \( 5 < \text{sal} < 10 \) are duplicated...

⇒ Prefer to deal with replication explicitly

Example: \( \mathcal{F} = \{ F_5, F_6, F_7 \} \)

\[ F_5 = \sigma_{\text{sal} \leq 5} E \quad \quad F_6 = \sigma_{5< \text{sal} <10} E \]

\[ F_7 = \sigma_{\text{sal} \geq 10} E \]

☞ Then replicate \( F_6 \) if convenient
(part of allocation problem)
Desired properties for horizontal fragmentation

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

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

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

3. Reconstruction
   \[ \bigcup F_i \]

---

<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>

How many fragments to generate? How to decide?
How do we get completeness and disjointness?

(1) Check it “manually”!

\[ F_1 = \sigma_{\text{sal} < 10} E \ ; \ F_2 = \sigma_{\text{sal} \geq 10} E \]

e.g., \[ F_1 = \sigma_{\text{sal} < 10} E \ ; \ F_2 = \sigma_{\text{sal} \geq 10} E \]

(2) “Automatically” generate fragments with these properties

Desired simple predicates \( \Rightarrow \) Fragments

Example of generation

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

1. \( A < 10 \land A > 5 \land \text{Loc}=S_A \land \text{Loc}=S_B \)
2. \( A < 10 \land A > 5 \land \text{Loc}=S_A \land \neg (\text{Loc}=S_B) \)
3. \( A < 10 \land A > 5 \land \neg (\text{Loc}=S_A) \land \text{Loc}=S_B \)
4. \( A < 10 \land A > 5 \land \neg (\text{Loc}=S_A) \land \neg (\text{Loc}=S_B) \)
5. \( A < 10 \land \neg (A > 5) \land \text{Loc}=S_A \land \text{Loc}=S_B \)
6. \( A < 10 \land \neg (A > 5) \land \text{Loc}=S_A \land \neg (\text{Loc}=S_B) \)
7. \( A < 10 \land \neg (A > 5) \land \neg (\text{Loc}=S_A) \land \text{Loc}=S_B \)
8. \( A < 10 \land \neg (A > 5) \land \neg (\text{Loc}=S_A) \land \neg (\text{Loc}=S_B) \)

A \leq 5

Minterm predicates (part II)

9. \( \neg (A < 10) \land A > 5 \land \text{Loc}=S_A \land \text{Loc}=S_B \)
10. \( \neg (A < 10) \land A > 5 \land \text{Loc}=S_A \land \neg (\text{Loc}=S_B) \)
11. \( \neg (A < 10) \land A > 5 \land \neg (\text{Loc}=S_A) \land \text{Loc}=S_B \)
12. \( \neg (A < 10) \land A > 5 \land \neg (\text{Loc}=S_A) \land \neg (\text{Loc}=S_B) \)
13. \( \neg (A < 10) \land \neg (A > 5) \land \text{Loc}=S_A \land \text{Loc}=S_B \)
14. \( \neg (A < 10) \land \neg (A > 5) \land \text{Loc}=S_A \land \neg (\text{Loc}=S_B) \)
15. \( \neg (A < 10) \land \neg (A > 5) \land \neg (\text{Loc}=S_A) \land \text{Loc}=S_B \)
16. \( \neg (A < 10) \land \neg (A > 5) \land \neg (\text{Loc}=S_A) \land \neg (\text{Loc}=S_B) \)

A \geq 10
Final fragments:

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

Note: elimination of useless fragments depends on application semantics:

e.g.: if LOC could be \(\neq S_A, \neq S_B\),
we need to add fragments

F4: \(5 < A < 10 \land \text{Loc} \neq S_A \land \text{Loc} \neq S_B\)
F8: \(A \leq 5 \land \text{Loc} \neq S_A \land \text{Loc} \neq S_B\)
F12: \(A \geq 10 \land \text{Loc} \neq S_A \land \text{Loc} \neq S_B\)
Summary

• Given simple predicates Pr= \{ p_1, p_2,.. p_m \}
minterm predicates are

M={m | m = \bigwedge_{p_k \in Pr} p_k^*, \ 1 \leq k \leq m } where p_k^* is p_k or is \neg p_k

• Fragments \sigma_m R for all m \in M are complete and disjoint

Which simple predicates should we use in Pr?

⇔ Desired property of Pr:
- minimality
- completeness
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 = { }  \( F_1 =\{ E \} \)
2. Pr = {LOC=Sa, LOC=Sb}  
   \( F_2 =\{ \sigma_{loc=Sa} E, \sigma_{loc=Sb} E \} \)
3. Pr = {LOC=Sa, LOC=Sb, Sal<10}  
   \( F_3 =\{ \sigma_{loc=Sa \land sal<10} E, \sigma_{loc=Sa \land sal\geq10} E, \sigma_{loc=Sb \land sal<10} E, \sigma_{loc=Sb \land sal\geq10} E \} \)
In other words:

$Q_a$: Select ... $\text{loc} = S_a$ ...

$Q_b$: Select ... $\text{loc} = S_b$ ...

$\mathcal{F}_2$ is good...
(not $\mathcal{F}_1$, $\mathcal{F}_3$)

Informal definition

Set of predicates $Pr$ is *complete* if for every $F_i \in \mathcal{F}[Pr]$, every $t \in F_i$ has equal probability of access by every (major) application.

Note: $\mathcal{F}[Pr]$ is fragmentation defined by minterm predicates generated by $Pr$. 
Back to example:

\[ F_1 \]

- \( \text{Qa: Select ... loc = Sa ...} \)
- \( \text{Qb: Select ... loc = Sb ...} \)
- Tuples here have higher probability of access
- Tuples here have lower probability of access
- So \( F_1 \) is not “good”...

\[ F_2 \]

- \( \text{Qa: Select ... loc = Sa ...} \)
- \( \text{Qb: Select ... loc = Sb ...} \)
- Tuples here have same probability of access
- So \( F_2 \) is “good”...
- So is \( F_3 \) ...

\[ F_3 \]
Is \( P_r \) complete a good thing?

**Informal definition**

Set of predicates \( P_r \) is *minimal* if no \( P_r' \subset P_r \) is complete
Back to example:

(1) Pr = { } \[ \times \]
(2) Pr = {LOC=Sa, LOC=Sb} \[ \checkmark \]
(3) Pr = {LOC=Sa, LOC=Sb, Sal<10} \[ \checkmark \]

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:

\[ E(\#, \text{NM, SAL, LOC}) \]

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

\[ J(\#, \text{DES, ...}) \]

Common query for project:

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

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

---

### E1

<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>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

(at Sa)

### E2

<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>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

(at Sb)

How to fragment \( J \)?

- On which attribute?
- How to do it? (by looking at \( J \) alone?)

### J

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>project 1</td>
</tr>
<tr>
<td>7</td>
<td>project 2</td>
</tr>
<tr>
<td>5</td>
<td>project 3</td>
</tr>
<tr>
<td>12</td>
<td>project 4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>#</td>
<td>NM</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
</tr>
<tr>
<td>5</td>
<td>Joe</td>
</tr>
<tr>
<td>8</td>
<td>Tom</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

(at Sa)

<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>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(at Sb)

<table>
<thead>
<tr>
<th>#</th>
<th>Des</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>project 1</td>
</tr>
<tr>
<td>5</td>
<td>project 3</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

J1 = J \bowtie E1

= \Pi_J (J \bowtie E1)

<table>
<thead>
<tr>
<th>#</th>
<th>Des</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>project 2</td>
</tr>
<tr>
<td>12</td>
<td>project 4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

J2 = J \bowtie E2

Semijoin: \bowtie

---

**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>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- But no # = 33 in E₁ or in E₂!

This J tuple will not be in J₁ or J₂
Fragmentation not complete

To get completeness …

- Need to enforce referential integrity constraint:
  join attr(#) of member relation
  ↓ (⊆)
  join attr(#) of owner relation
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>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
E1
```

```
<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>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
E2
```

```
<table>
<thead>
<tr>
<th>#</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
```
\[
\begin{align*}
R[T] \Rightarrow & \quad R_1[T_1] \quad T_i \subseteq T \\
& \quad \vdots \\
& \quad R_n[T_n]
\end{align*}
\]

Just like normalization of relations

\textbf{Properties:} \quad R[T] \Rightarrow R_i[T_i]

(1) Completeness

\[
\bigcup_{\text{all } i} T_i = T
\]
(2) Disjointness
\[ T_i \cap T_j = \emptyset \text{ for all } i, j \neq j \]

\[ E(\#, \text{LOC}, \text{SAL}) \]

\[ E_1(\#, \text{LOC}) \]

\[ E_2(\text{SAL}) \]

Not a desirable property!!
(could not reconstruct R!)

---

(3) Lossless join
\[ \forall i \quad R_i = R \]

One way to achieve lossless join:
Repeat key in all fragments, i.e.,
\[ \text{Key} \subseteq T_i \text{ for all } i \]
How do we decide what attributes are grouped with which?

Example:
E(#,NM,LOC,SAL) →
E1(#,NM,LOC) →
E2(#,SAL) →
E1(#,NM) →
E2(#,LOC) →
E3(#,SAL)

Attribute affinity matrix

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>45</td>
<td>48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>75</td>
<td></td>
</tr>
</tbody>
</table>

Aff(Ai, Aj) captures the access frequencies of applications for attributes Ai and Aj.

• Ref 1 (Ozsu & Valduriez, Chap 5) discusses
  - How to build affinity matrix (What about 3 or higher dimensions?)
  - How to identify attribute clusters
  - How to partition relation
• You are not responsible for
  - Clustering and partitioning algorithms
    (i.e., Skip pages 135-145)

Allocation

Example: $E(\#, NM, LOC, SAL) \Rightarrow$

$F_1 = \sigma_{loc=Sa} E ; F_2 = \sigma_{loc=Sb} E$

$Q_a$: select ... where $loc=Sa$...
$Q_b$: 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
  - …
**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 showing a network of nodes and edges with labels $C_{i,1}$, $C_{i,2}$, and $C_{i,3}$, and a stream of read requests for $F$ at $t_i$ REQ/SEC](image)
Write cost

\[ \sum_{i=1}^{m} \sum_{j=1}^{m} X_j \ u_i \ C'_{ij} \]

- \( i \): Originating site of request
- \( j \): Site being updated
- \( X_j \): 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**

- \( F \) updates/sec
- \( F \) updates/sec
- \( F \) updates/sec
- \( ui \) updates/sec

---
Storage cost:

\[
\sum_{i=1}^{m} X_i \ di
\]

\(X_i: \begin{cases} 
0 & \text{if } F \text{ not stored at } S_i \\
1 & \text{if } F \text{ stored at } S_i 
\end{cases}
\)

di: storage cost at Si

Target function

\[
\min \left\{ \sum_{i=1}^{m} [t_i \times \text{MIN } C_{ij} + \sum_{j=1}^{m} X_j \times u_i \times C'_{ij}] + \sum_{i=1}^{m} X_i \times di \right\}
\]
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 $5+5+10+10\times2 = 40$.
  – 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