

CS 347:  
Distributed Databases and  
Transaction Processing  
**Notes02: Distributed DB Design**

Hector Garcia-Molina  
Zoltan Gyongyi

# Distributed DB Design

Chapter 5  
Ozsu & Valduriez

Top-down approach: - have DB...  
- how to split and allocate the sites

Multi-DBs (or bottom-up): - no design issues

## Two issues in DDB design:

- Fragmentation
- Allocation

Note: issues not independent,  
but will cover separately

# Example

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

40% of queries:

$Q_A$ : select \*  
from E  
where loc =  $S_A$   
and ...

40% of queries:

$Q_B$ : select \*  
from E  
where loc =  $S_B$   
and ...

# Example

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

40% of queries:

$Q_A$ : select \*  
from E  
where loc =  $S_A$   
and ...

40% of queries:

$Q_B$ : select \*  
from E  
where loc =  $S_B$   
and ...

Motivation: Two sites:  $S_A, S_B$

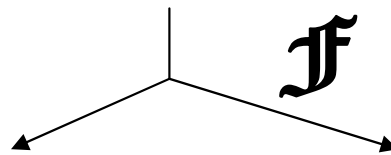


- It does not take a rocket scientist to figure out fragmentation...

# NM Loc Sal

E

5	Joe	S <sub>A</sub>	10
7	Sally	S <sub>B</sub>	25
8	Tom	S <sub>A</sub>	15
:			:



# NM Loc Sal

5	Joe	S <sub>A</sub>	10
8	Tom	S <sub>A</sub>	15
:			

At S<sub>A</sub>

# NM Loc Sal

7	Sally	S <sub>B</sub>	25
:			

At S<sub>B</sub>

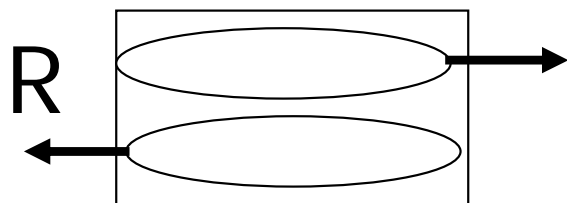
$$\mathcal{F} = \{ F_1, F_2 \}$$

$$F_1 = \sigma_{\text{loc}=S_A} E \quad F_2 = \sigma_{\text{loc}=S_B} E$$

$\Rightarrow$  called primary horizontal fragmentation

# Fragmentation

- Horizontal



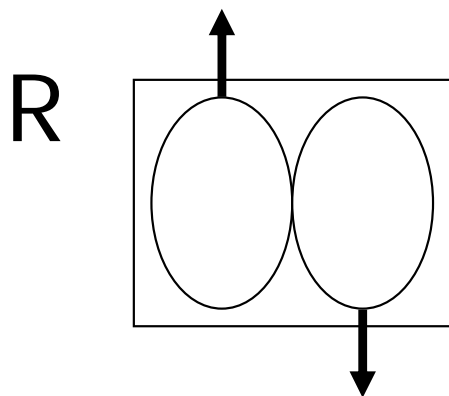
Primary

depends on local attributes

Derived

depends on foreign relation

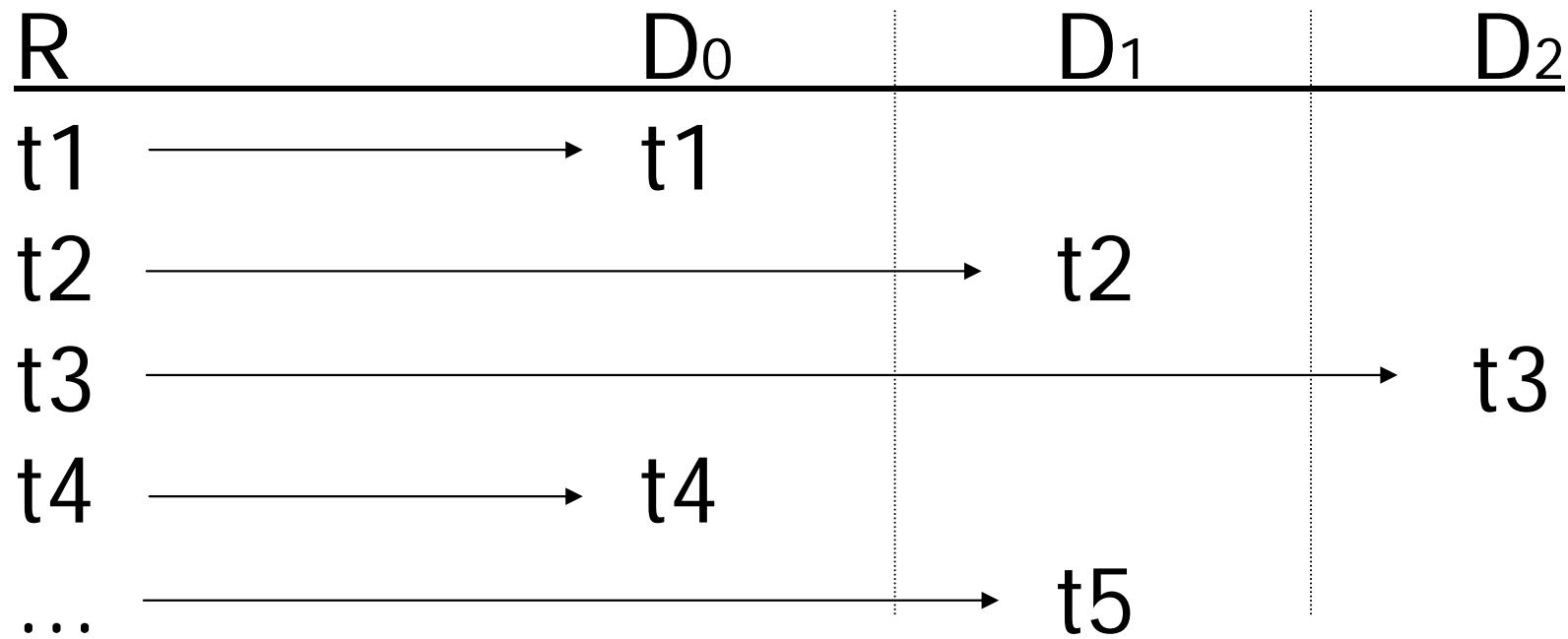
- Vertical



# Three common horizontal partitioning techniques

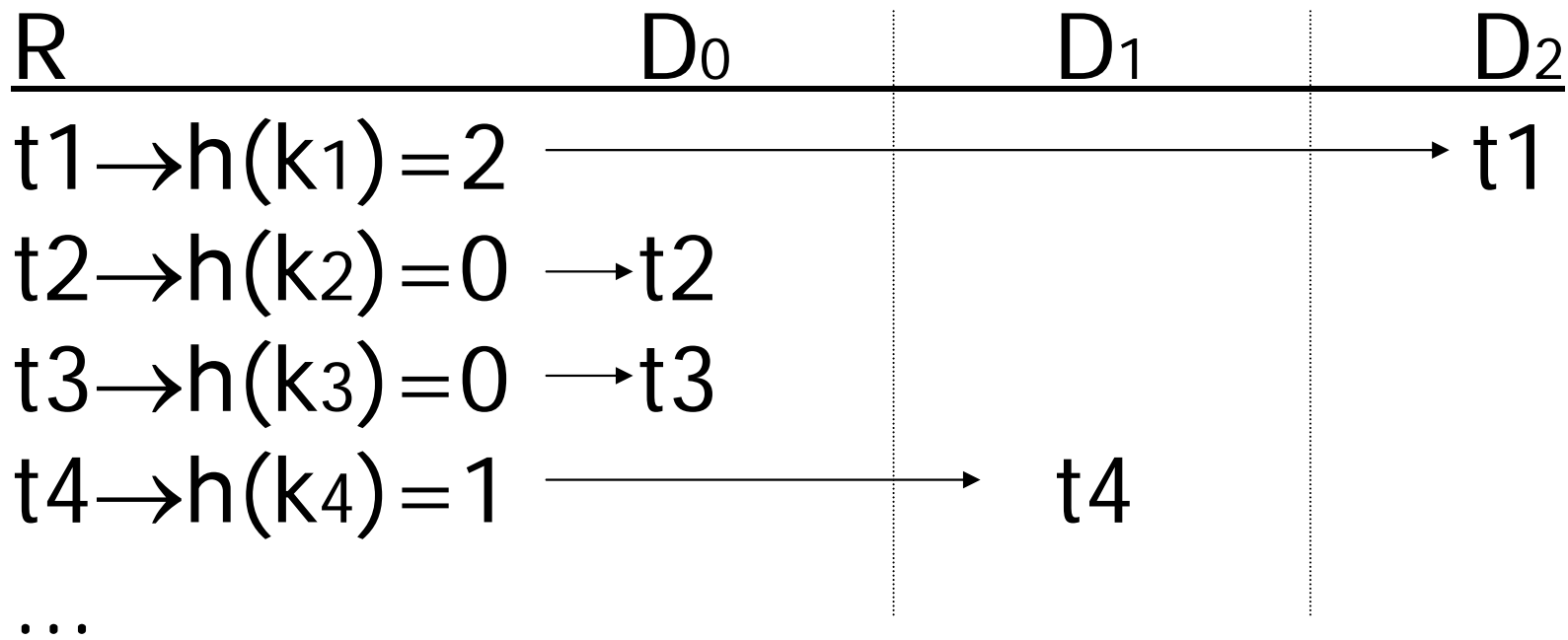
- Round robin
- Hash partitioning
- Range partitioning

- Round robin



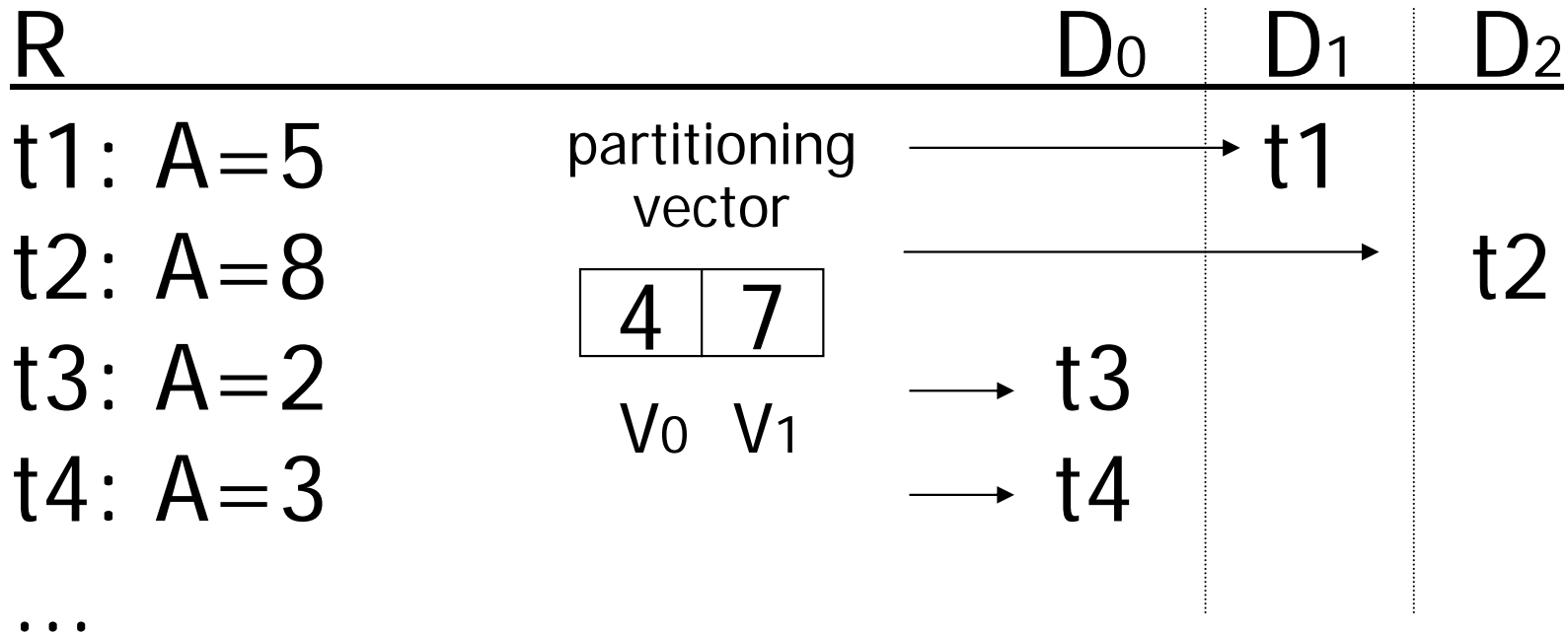
- Evenly distributes data
- Good for scanning full relation
- Not good for point or range queries

- Hash partitioning



- Good for point queries on key; also for joins
- Not good for range queries; point queries not on key
- If hash function good, even distribution

- Range partitioning



- Good for some range queries on A
- Need to select good vector: else unbalance
  - data skew
  - execution skew

# Which are good fragmentations?

Example:

$$\mathcal{F} = \{ F_1, F_2 \}$$

$$F_1 = \sigma_{\text{sal} < 10} E$$

$$F_2 = \sigma_{\text{sal} > 20} E$$

# Which are good fragmentations?

Example:

$$\mathcal{F} = \{ F_1, F_2 \}$$

$$F_1 = \sigma_{\text{sal} < 10} E$$

$$F_2 = \sigma_{\text{sal} > 20} E$$

☒ Problem: Some tuples lost!

# Which are good fragmentations?

Second example:

$$\mathcal{F} = \{ F_3, F_4 \}$$

$$F_3 = \sigma_{sal < 10} E$$

$$F_4 = \sigma_{sal > 5} E$$

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

# Prefer to deal with replication explicitly

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

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

$$F_7 = \sigma_{\text{sal} \geq 10} E$$

- ➔ Then replicate  $F_6$  if convenient (part of allocation problem)

# Desired properties for horizontal fragmentation

$$R \Rightarrow \mathcal{F} = \{ F_1, F_2, \dots \}$$

## (1) Completeness

$$\forall t \in R, \exists F_i \in \mathcal{F} \text{ such that } t \in F_i$$

## (2) Disjointness

$\forall t \in F_i, \neg \exists F_j$  such that  
 $t \in F_j, i \neq j, F_i, F_j \in \mathcal{F}$

# How do we get completeness and disjointness?

(1) Check it “manually”!

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

# How do we get completeness and disjointness?

(2) “Automatically” generate fragments with these properties

Desired simple predicates  $\Rightarrow$  Fragments

# Example of generation

- Say queries use predicates:  
 $A < 10, A > 5, \text{Loc} = S_A, \text{Loc} = S_B$
- Next: - generate "minterm" predicates  
- eliminate useless ones

# Minterm predicates (part I)

- (1)  $A < 10 \wedge A > 5 \wedge LOC = S_A \wedge LOC = S_B$
- (2)  $A < 10 \wedge A > 5 \wedge LOC = S_A \wedge \neg(LOC = S_B)$
- (3)  $A < 10 \wedge A > 5 \wedge \neg(LOC = S_A) \wedge LOC = S_B$
- (4)  $A < 10 \wedge A > 5 \wedge \neg(LOC = S_A) \wedge \neg(LOC = S_B)$
- (5)  $A < 10 \wedge \neg(A > 5) \wedge LOC = S_A \wedge LOC = S_B$
- (6)  $A < 10 \wedge \neg(A > 5) \wedge LOC = S_A \wedge \neg(LOC = S_B)$
- (7)  $A < 10 \wedge \neg(A > 5) \wedge \neg(LOC = S_A) \wedge LOC = S_B$
- (8)  $A < 10 \wedge \neg(A > 5) \wedge \neg(LOC = S_A) \wedge \neg(LOC = S_B)$

$$5 < A < 10$$

## Minterm predicates (part I)

- (1)  ~~$A < 10 \wedge A > 5 \wedge \text{LOC} = S_A \wedge \text{LOC} = S_B$~~
- (2)  $A < 10 \wedge A > 5 \wedge \text{LOC} = S_A \wedge \neg(\text{LOC} = S_B)$
- (3)  $A < 10 \wedge A > 5 \wedge \neg(\text{LOC} = S_A) \wedge \text{LOC} = S_B$
- (4)  ~~$A < 10 \wedge A > 5 \wedge \neg(\text{LOC} = S_A) \wedge \neg(\text{LOC} = S_B)$~~
- (5)  ~~$A < 10 \wedge \neg(A > 5) \wedge \text{LOC} = S_A \wedge \text{LOC} = S_B$~~
- (6)  $A < 10 \wedge \neg(A > 5) \wedge \text{LOC} = S_A \wedge \neg(\text{LOC} = S_B)$
- (7)  $A < 10 \wedge \neg(A > 5) \wedge \neg(\text{LOC} = S_A) \wedge \text{LOC} = S_B$
- (8)  ~~$A < 10 \wedge \neg(A > 5) \wedge \neg(\text{LOC} = S_A) \wedge \neg(\text{LOC} = S_B)$~~

$$A \leq 5$$

## Minterm predicates (part II)

- (9)  $\neg(A < 10) \wedge A > 5 \wedge LOC = S_A \wedge LOC = S_B$
- (10)  $\neg(A < 10) \wedge A > 5 \wedge LOC = S_A \wedge \neg(LOC = S_B)$
- (11)  $\neg(A < 10) \wedge A > 5 \wedge \neg(LOC = S_A) \wedge LOC = S_B$
- (12)  $\neg(A < 10) \wedge A > 5 \wedge \neg(LOC = S_A) \wedge \neg(LOC = S_B)$
- (13)  $\neg(A < 10) \wedge \neg(A > 5) \wedge LOC = S_A \wedge LOC = S_B$
- (14)  $\neg(A < 10) \wedge \neg(A > 5) \wedge LOC = S_A \wedge \neg(LOC = S_B)$
- (15)  $\neg(A < 10) \wedge \neg(A > 5) \wedge \neg(LOC = S_A) \wedge LOC = S_B$
- (16)  $\neg(A < 10) \wedge \neg(A > 5) \wedge \neg(LOC = S_A) \wedge \neg(LOC = S_B)$

## Minterm predicates (part II)

- (9)  ~~$\neg(A < 10) \wedge A > 5 \wedge \text{Loc} = S_A \wedge \text{Loc} = S_B$~~
- (10)  $\neg(A < 10) \wedge A > 5 \wedge \text{Loc} = S_A \wedge \neg(\text{Loc} = S_B)$
- (11)  $\neg(A < 10) \wedge A > 5 \wedge \neg(\text{Loc} = S_A) \wedge \text{Loc} = S_B$
- (12)  ~~$\neg(A < 10) \wedge A > 5 \wedge \neg(\text{Loc} = S_A) \wedge \neg(\text{Loc} = S_B)$~~
- (13)  ~~$\neg(A < 10) \wedge \neg(A > 5) \wedge \text{Loc} = S_A \wedge \text{Loc} = S_B$~~
- (14)  ~~$\neg(A < 10) \wedge \neg(A > 5) \wedge \text{Loc} = S_A \wedge \neg(\text{Loc} = S_B)$~~
- (15)  ~~$\neg(A < 10) \wedge \neg(A > 5) \wedge \neg(\text{Loc} = S_A) \wedge \text{Loc} = S_B$~~
- (16)  ~~$\neg(A < 10) \wedge \neg(A > 5) \wedge \neg(\text{Loc} = S_A) \wedge \neg(\text{Loc} = S_B)$~~

$A \geq 10$

## Final fragments:

F<sub>2</sub>:  $5 < A < 10$       $\wedge$      LOC = S<sub>A</sub>

F<sub>3</sub>:  $5 < A < 10$       $\wedge$      LOC = S<sub>B</sub>

F<sub>6</sub>:  $A \leq 5$       $\wedge$      LOC = S<sub>A</sub>

F<sub>7</sub>:  $A \leq 5$       $\wedge$      LOC = S<sub>B</sub>

F<sub>10</sub>:  $A \geq 10$       $\wedge$      LOC = S<sub>A</sub>

F<sub>11</sub>:  $A \geq 10$       $\wedge$      LOC = S<sub>B</sub>

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:  $5 < A < 10$        $\wedge$      $LOC \neq S_A \wedge LOC \neq S_B$

F8:  $A \leq 5$        $\wedge$      $LOC \neq S_A \wedge LOC \neq S_B$

F12:  $A \geq 10$        $\wedge$      $LOC \neq S_A \wedge LOC \neq S_B$

## Why does this work?

Predicates:  $p1 \wedge p2 \wedge p3 \wedge p4$

$p1 \wedge p2 \wedge p3 \wedge \neg p4$

$\vdots$

$\neg p1 \wedge \neg p2 \wedge \neg p3 \wedge \neg p4$

## (1) Completeness:

Take  $t \in R$ ,  $p_i(t)$  must be T or F!

Say  $p_1(t) = T$ ,  $p_2(t) = T$ ,  $p_3(t) = F$ ,  $p_4(t) = F$

Then  $t$  is in fragment with predicate

$$p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4$$

## (2) Disjointness

Say  $t \in$  fragment  $p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4$

Then

$$p_1(t) = T, \quad p_2(t) = T, \quad p_3(t) = F, \quad p_4(t) = F$$

$\Rightarrow$   $t$  cannot be in any other fragment!

# Summary

- Given simple predicates  $P_r = \{ p_1, p_2, \dots, p_m \}$   
minterm predicates are

$$M = \{ m \mid m = \bigwedge_{p_k \in P_r} p_k^*, \quad 1 \leq k \leq m \}$$

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!



## Informal definition

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

Note:  $\mathcal{F}[P_r]$  is fragmentation defined by  
minterm predicates generated by  $P_r$

## Informal definition

Set of predicates  $Pr$  is minimal if no  $Pr' \subset Pr$  is complete

# Summary: horizontal fragmentation

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

# Vertical fragmentation

Example:

E

#	NM	Loc	Sal
5	Joe	Sa	10
7	Sally	Sb	25
8	Fred	Sa	15
...			

E1

#	NM	Loc
5	Joe	Sa
7	Sally	Sb
8	Fred	Sa
...		

E2

#	Sal
5	10
7	25
8	15
...	

$$R[T] \Rightarrow \begin{array}{l} R_1[T_1] \\ \vdots \\ R_n[T_n] \end{array} \quad T_i \subseteq T$$

☒ Just like normalization of relations

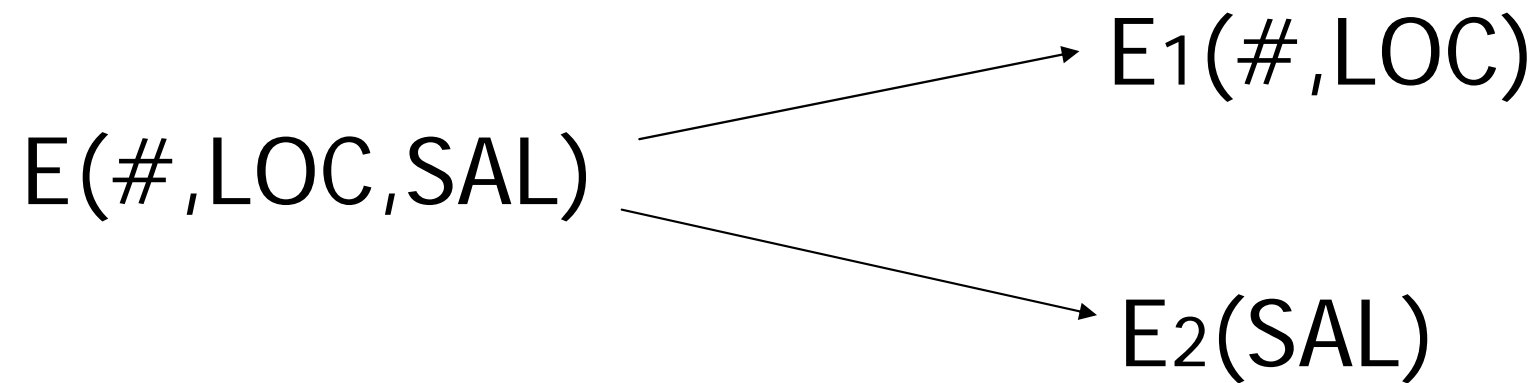
Properties:  $R[T] \Rightarrow R_i[T_i]$

(1) Completeness

$$\bigcup_{\text{all } i} T_i = T$$

## (2) Disjointness

$T_i \cap T_j = \emptyset$  for all  $i, j \ i \neq j$



## (2) Disjointness

$T_i \cap T_j = \emptyset$  for all  $i, j \ i \neq j$

$E(\#, LOC, SAL)$

$E_1(\#, LOC)$

$E_2(SAL)$

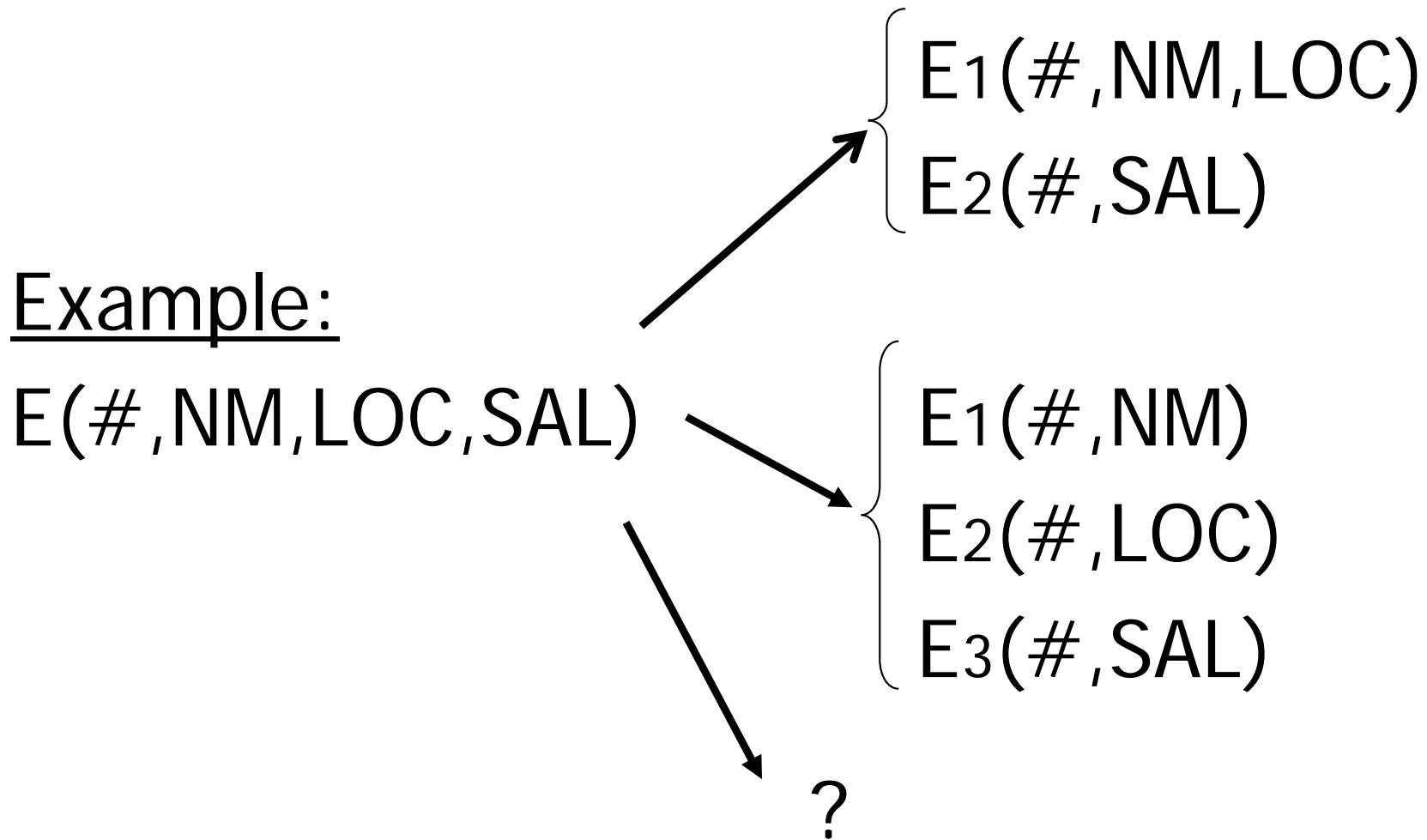
Not a desirable property!  
(could not reconstruct R)

### (3) Lossless join

$$\bigotimes_{\text{all } i} R_i = R$$

➔ One way to achieve lossless join:  
repeat key in all fragments, i.e.,  
 $\text{key} \subseteq T_i$  for all  $i$

⇔ How do we decide what attributes are grouped with which?



# Attribute affinity matrix

	A1	A2	A3	A4	A5
A1	-	-	-	-	-
A2	50	-	-	-	-
A3	45	48	-	-	-
A4	1	2	0	-	-
A5	0	0	4	75	-

# Attribute affinity matrix

	A1	A2	A3	A4	A5
A1	-	-	-	-	-
A2	50	-	-	-	-
A3	45	48	-	-	-
A4	1	2	0	-	-
A5	0	0	4	75	-

$R_1[K, A_1, A_2, A_3]$

$R_2[K, A_4, A_5]$

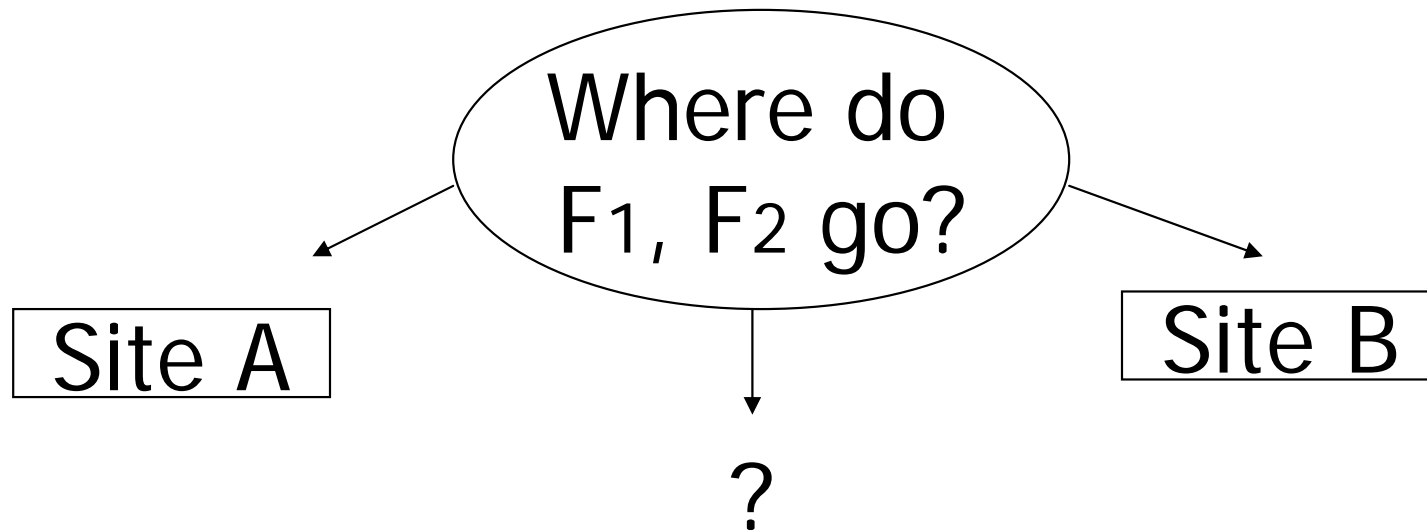
# Allocation

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

$$F_1 = \sigma_{loc=S_A} E ; F_2 = \sigma_{loc=S_B} E$$

$Q_A$ : select ... where  $loc=S_A$  ...

$Q_B$ : select ... where  $loc=S_B$  ...



# 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?

## More issues

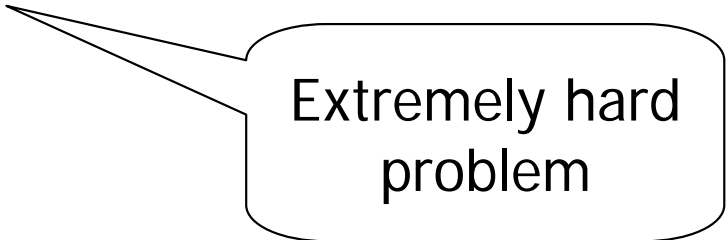
- What is query processing strategy?
  - How are joins done?
  - Where are answers collected?

# 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
  - ...



Extremely hard problem

## Example: Single fragment F

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

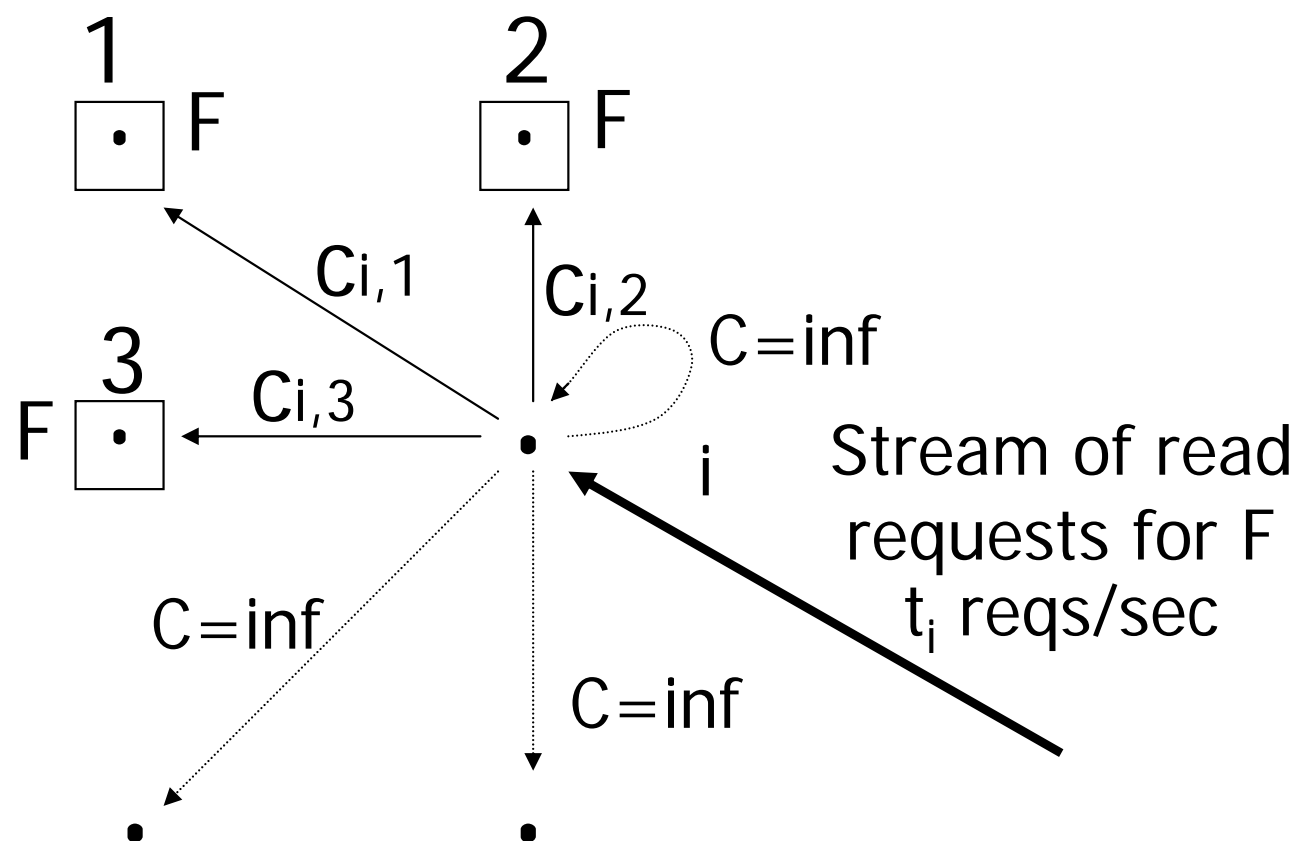
$i$ : Originating site of request (1...m)

$t_i$ : Read traffic at  $S_i$

$C_{ij}$ : Retrieval cost

Accessing fragment F at  $S_j$  from  $S_i$

# Scenario - Read cost



## *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

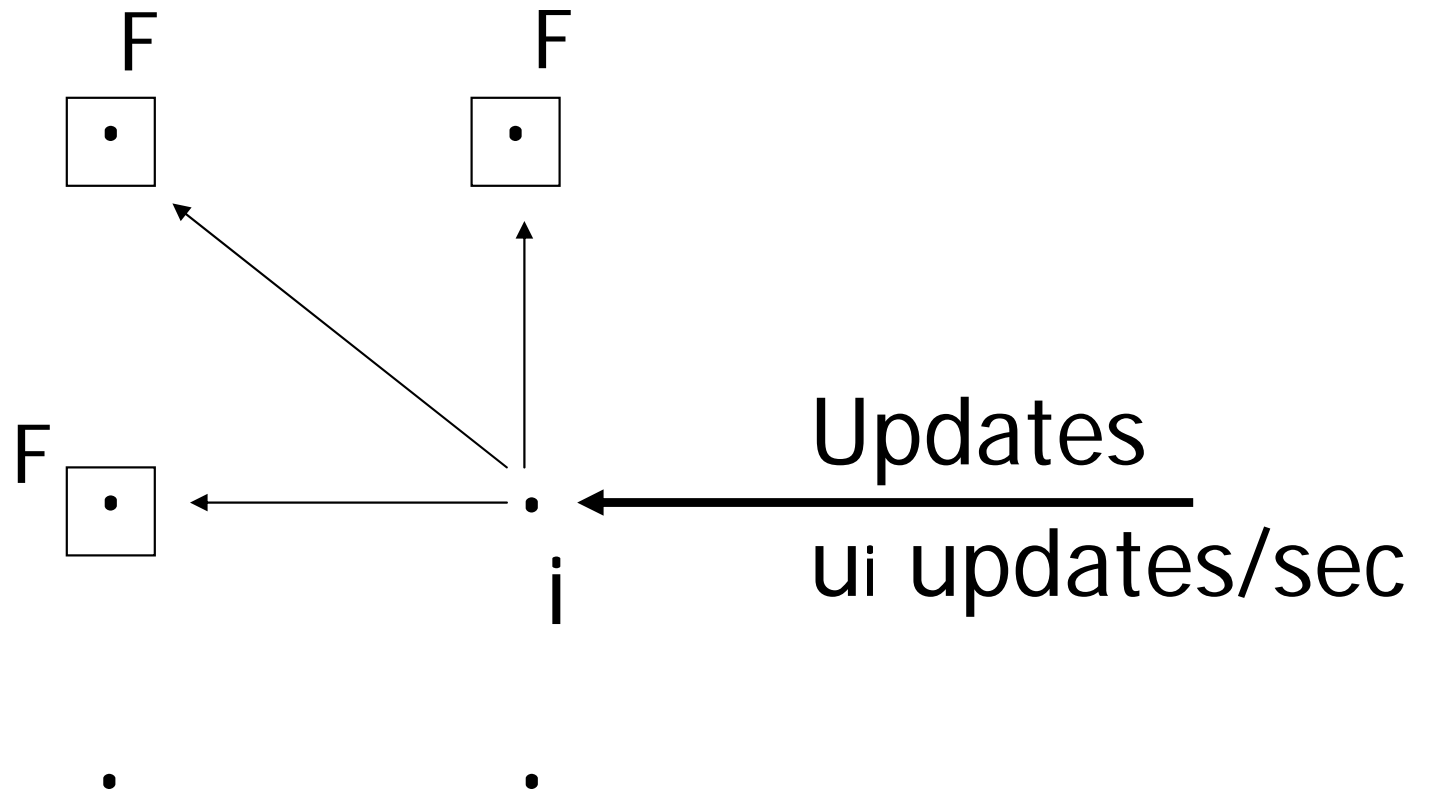
$X_j$ :  $\begin{cases} 0 & \text{if } F \text{ not stored at } S_j \\ 1 & \text{if } F \text{ stored at } S_j \end{cases}$

$u_i$ : Write traffic at  $S_i$

$C'_{ij}$ : Write cost

Updating  $F$  at  $S_j$  from  $S_i$

# Scenario - Write cost



Storage cost:

$$\sum_{i=1}^m X_i d_i$$

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

$d_i$ : storage cost at  $S_i$

# Target function:

$$\min \left\{ \sum_{i=1}^m [t_i \times \min_j C_{ij}] + \sum_{j=1}^m X_j \times u_i \times C'_{ij} \right. \\ \left. + \sum_{i=1}^m X_i \times d_i \right\}$$

# Can add more complications:

Examples:

- Multiple fragments
- Fragment sizes
- Concurrency control cost

# Summary

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