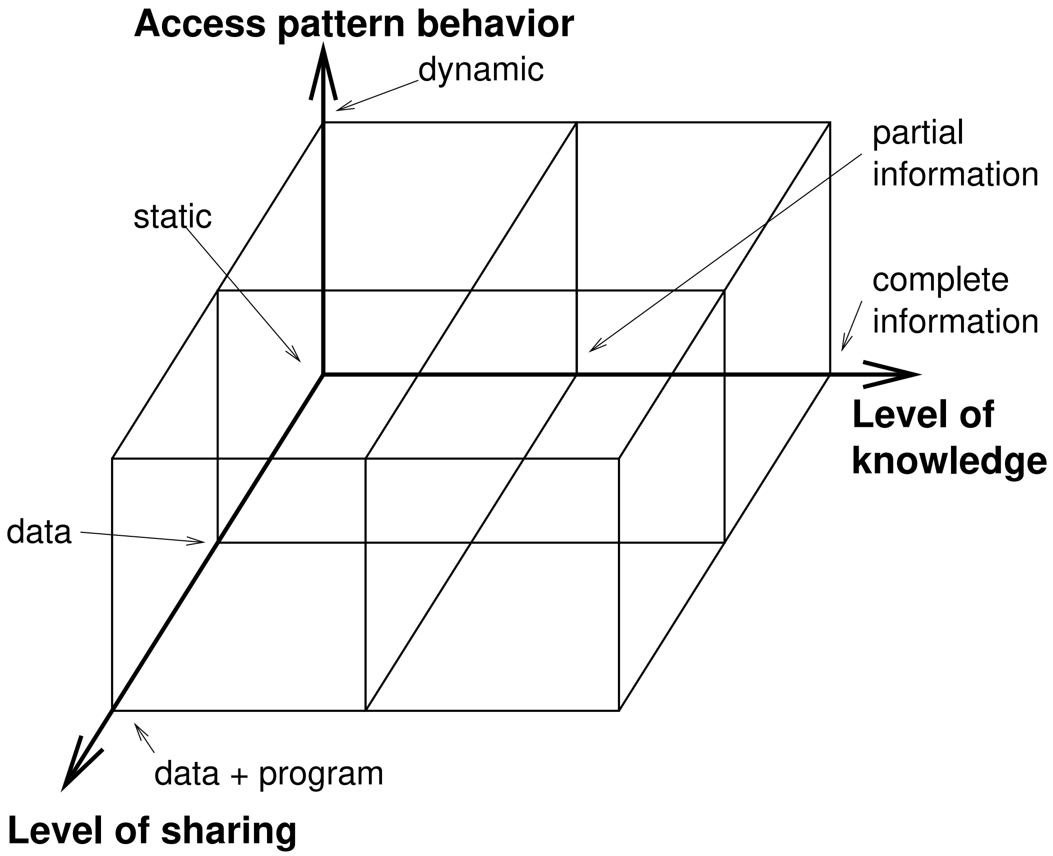
DDBMS DESIGN

Design Problem

- general setting
  - making decisions about the placement of $data$ and $programs$ across the sites of a computer network as well as possiby designing the network itself
- in DDBMS, the placement of applications entails
  - placement of the distributed DBMS software
  - placement of the applications that run on the database

DIMENSIONS OF THE PROBLEM



Level of sharing: none vs share data or programs vs share data & programs

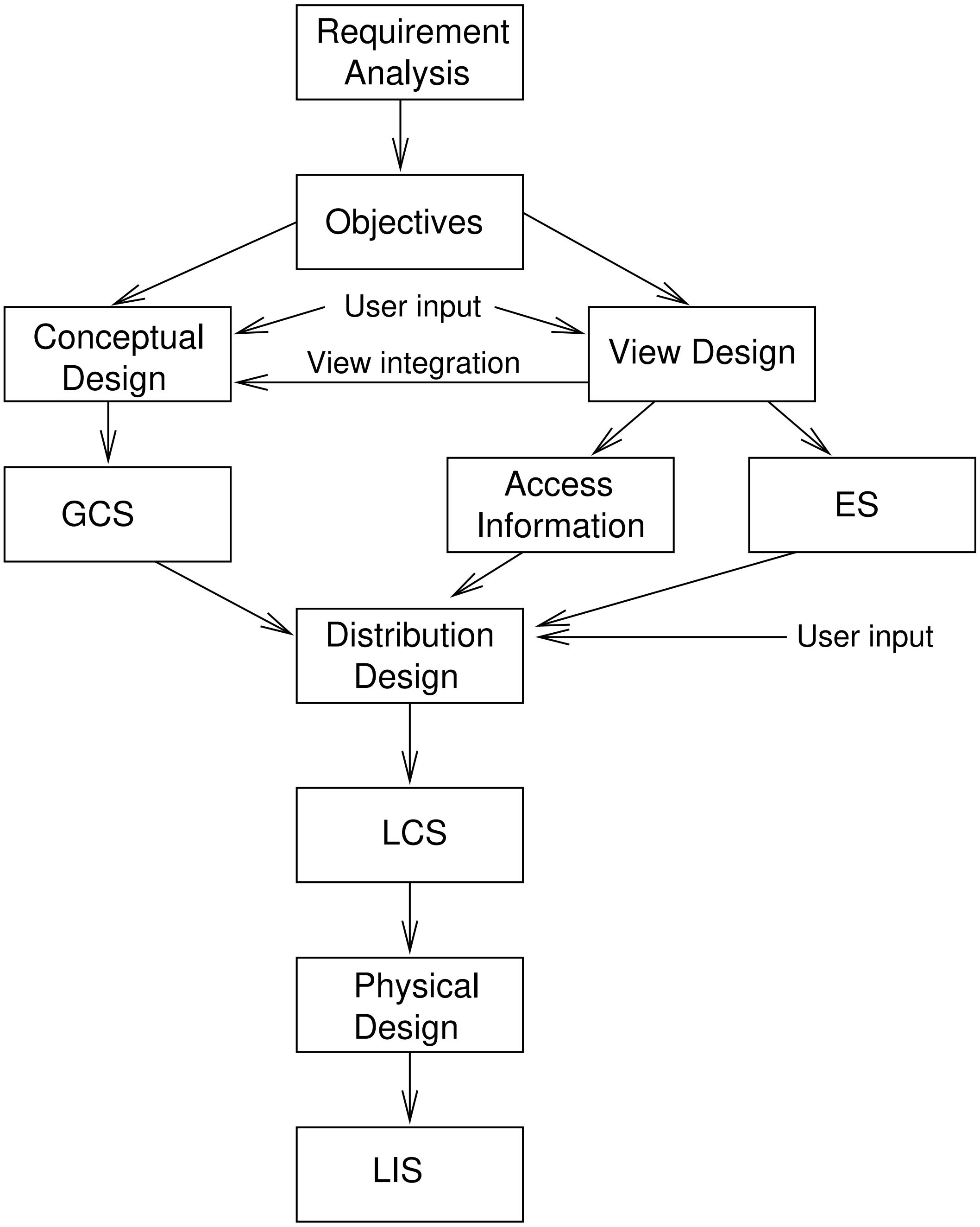
Access pattern behavior: static vs dynamic

Level of knowledge: (data access pattern behavior) none vs partial information vs complete information

DISTRIBUTION DESIGN

- top-down
  - mostly in designing systems from scratch
  - mostly in homogeneous system
- bottom-up
  - when the databases already exist at a number of sites

TOP-DOWN DESIGN



ES: External Schema definitions

GCS: Global Conceptual Schema

LCS: Local Conceptual Schema

LIS: Local Internal Schema

DISTRIBUTION DESIGN ISSUES

- why fragment at all?
- how to fragment?
- how much to fragment?
- how to test correctness?
- how to allocate?
- information requirements?

FRAGMENTATION

- can't we just distribute relations?
- what is a reasonable unit of distribution?
  - relation
    - views are subsets of relations
    - extra communication cost
- fragments of relations (sub-relations)
  - concurrent execution of a number of transactions that access different portions of a relation
  - views that cannot be defined on a single fragment will require extra processing
  - semantic data control (such as integrity enforcement) becomes more difficult

FRAGMENTATION ALTERNATIVES

Horizontal Fragmentation

Original relation:

JNO	JNAME	BUDGET	LOC
J1	Instrumentation	150,000	Montreal
J2	Database Development	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	310,000	Paris
J5	CAD/CAM	500,000	Boston

Projects with budgets less than $200,000
Horizontal fragment 1:

JNO	JNAME	BUDGET	LOC
J1	Instrumentation	150,000	Montreal
J2	Database Development	135,000	New York

Projects with budgets greater than $200,000
Horizontal fragment 2:

JNO	JNAME	BUDGET	LOC
J3	CAD/CAM	250,000	New York
J4	Maintenance	310,000	Paris
J5	CAD/CAM	500,000	Boston

FRAGMENTATION ALTERNATIVES (CONT)

Vertical Fragmentation

JNO	JNAME	BUDGET	LOC
J1	Instrumentation	150,000	Montreal
J2	Database Development	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	310,000	Paris
J5	CAD/CAM	500,000	Boston

information about project budgets

JNO	BUDGET
J1	150,000
J2	135,000
J3	250,000
J4	310,000
J5	500,000

information about project names and locations

JNO	JNAME	LOC
J1	Instrumentation	Montreal
J2	Database Development	New York
J3	CAD/CAM	New York
J4	Maintenance	Paris
J5	CAD/CAM	Boston

Note: Key is included in all the vertical fragments.

DEGREE OF FRAGMENTATION

- two extremes
  - tuples or attributes
  - relations
- finite number of alternatives
- find the suitable level of partitioning within this range

COMPLETENESS OF FRAGMENTATION

relation R is decomposed into fragments $R_1,\ R_2,\ \ldots,\ R_n$

- **Completeness**
  - decomposition is complete iff each data item in R can also be found in some R_i
- **Reconstruction**
  - there should exist some relational operation ∇ such that
$$R = \nabla_{1 \leq i \leq n} R_i$$
- **Disjointness**
  - if data item d_i is in R_j, then d_i should not be in any other fragment R_k ($k \neq j$).

ALLOCATION ALTERNATIVES

- **Non-replicated**
  - partitioned: each fragment resides at only one site
- **Replicated**
  - fully replicated: each fragment at each site
  - partially replicated: each fragment at some of the sites
- **Rule of thumb**:

  $$\text{if } \frac{read - only\ queries}{update\ queries} \geq 1,$$

  replication is advantageous; otherwise replication may cause problems

COMPARISON OF REPLICATION ALTERNATIVES

	Full Replication	Partial Replication	Partitioning
Query Processing	easy	same difficulty	
Directory Management	easy or non-existent	same difficulty	
Concurrency Management	moderate	difficult	easy
Reliability	very high	high	low
Reality	possible application	realistic	possible application

INFORMATION REQUIREMENTS

- four categories
  - database information
  - application information
  - communication network information
  - computer system information

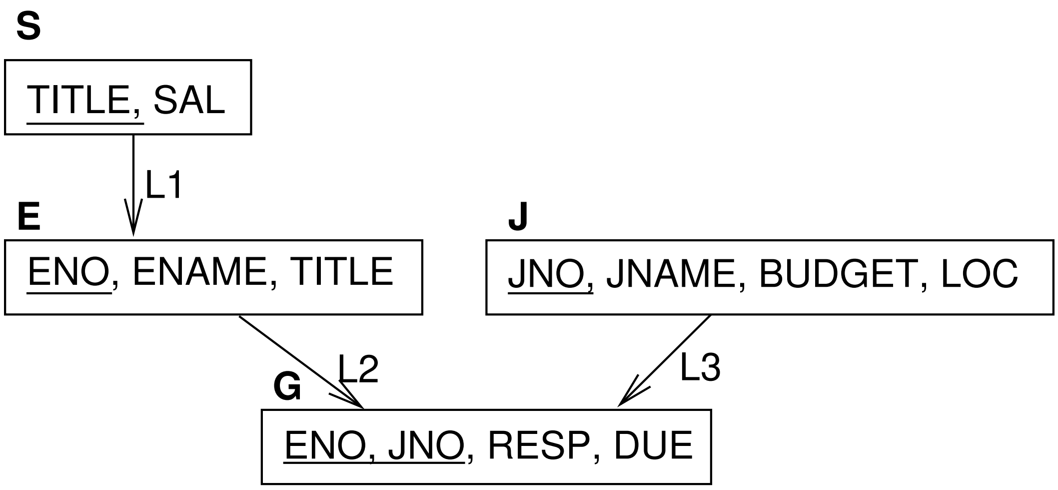
FRAGMENTATION

- Horizontal Fragmentation
  - primary horizontal fragmentation (PHF)
  - derived horizontal fragmentation (DHF)
- Vertical Fragmentation (VF)
- Hybrid Fragmentation (HF)

PHF – INFORMATION REQUIREMENTS

Database Information

- relationship



- cardinality of each relation: $card(R)$
- L1, L2 and L3 are links.
  owner(L1)=S
  member(L1)=E

PHF – INFORMATION REQUIREMENTS (CONT)

Application Information

- $simple\,predicates$: Given $R[A_1, A_2, \ldots, A_n]$, a simple predicate p_j is

  $$p_j : A_i\,\theta\,Value$$

  where $\theta \in \{=, <, \leq, >, \geq, \neq\},\ Value \in D_i$, and D_i is the domain of A_i.

- for relation R with k simple predicates, we define $Pr = \{p_1, p_2, \ldots, p_k\}$, for example,

  JNAME = "Maintenance"
  BUDGET $\leq$ 200,000

- $minterm\,predicates$: Given R and $Pr = \{p_1, \ldots, p_k\}$, define $M = \{m_1, m_2, \ldots, m_s\}$ as

  $$M = \{m_i | m_i = \wedge_{P_j \in Pr} P_j*\},\ 1 \leq j \leq k,\ 1 \leq i \leq s$$

  where $P_j* = P_j$ or $P_j* = \neg(P_j)$

  – Note: A minterm predicate is a conjunction of simple predicates and negative predicates.

PHF – INFORMATION REQUIREMENTS (CONT)

Example

m_1 : JNAME="Maintenance" $\wedge$ BUDGET $\leq$ 200,000

m_2 : NOT(JNAME="Maintenance") $\wedge$ BUDGET $\leq$ 200,000

m_3 : JNAME="Maintenance" $\wedge$ NOT(BUDGET $\leq$ 200,000)

m_4 : NOT(JNAME="Maintenance") $\wedge$ NOT(BUDGET $\leq$ 200,000)

PHF – INFORMATION REQUIREMENTS (CONT)

Application Information (Cont)

- $minterm\ selectivities : sel(m_i)$
  - the number of tuples of the relation that would be accessed by a user query which is specified according to a given minterm predicate m_i
- $access\ frequencies : acc(q_j)$
  - the frequency with which a user application q_j accesses data in a given period of time
  - access frequency for a minterm predicate m_i can also be defined, denoted as $acc(m_i)$

PRIMARY HORIZONTAL FRAGMENTATION

- Definition:

$$R_j = \sigma_{F_j}(R), \quad 1 \leq j \leq w$$

  where F_j is a selection formula, which is (preferably) a minterm predicate

- a horizontal fragment R_i of relation R consists of all the tuples of R which satisfy a minterm predicate m_i
- given a set of minterm predicates M, there are as many horizontal fragments of relation R as there are minterm predicates
- set of horizontal fragments also referred to as $minterm\ fragments$

PHF – ALGORITHM

Given: A relation R, the set of simple predicates Pr
Output: The set of fragments of $R = \{R_1, R_2, \ldots, R_w\}$ which obey the fragmentation rules

- Pr should be $complete$
- Pr should be $minimal$

COMPLETENESS OF SIMPLE PREDICATES

- Pr is $complete$ iff the accesses to the tuples of the minterm fragments defined on Pr requires that two tuples of the same minterm fragment have the same probability of being accessed by any application, for example
  - assume J[JNO,JNAME,BUDGET,LOC] has two applications define on it
  - find the budgets of projects at each location (1)
  - find projects with budgets less than $200,000 (2)
- According to (1),

$$Pr = \{LOC = \text{``}Montreal'', LOC = \text{``}New\,York'', LOC = \text{``}Paris'', LOC = \text{``}Boston''\}$$

  which is not complete with respect to (2)
- Modify

$$Pr = \{LOC = \text{``}Montreal'', LOC = \text{``}New\,York'', LOC = \text{``}Paris'', LOC = \text{``}Boston'', BUDGET \leq 200000, BUDGET > 200000\}$$

  which is complete

MINIMALITY OF SIMPLE PREDICATES

- if a predicate causes a fragment f to be further fragmented into f_i and f_j, then there should be at least one application that accesses f_i and f_j differently
- i.e., the simple predicate should be $relevant$ in determining a fragmentation
- if all the predicates of a set Pr are relevant, then Pr is $minimal$
- Let m_i and m_i' 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_i' contains $\neg p_i$. Let f_i and f_i' be two fragments defined according to m_i and m_i'. Then p_i is relevant if and only if

$$\frac{acc(m_i)}{card(f_i)} \neq \frac{acc(m_i')}{card(f_i')}$$

MINIMALITY OF SIMPLE PREDICATES (CONT)

Example:

$$Pr = \{LOC = \text{“}Montreal'', LOC = \text{“}New\,York'', \\ LOC = \text{“}Paris'', LOC = \text{“}Boston'', \\ BUDGET \leq 200000, BUDGET > 200000\}$$

is minimal (in addition to being complete)

However, if we add
JNAME = “Instrumentation”
then Pr is NOT minimal since the new predicate is not relevant with respect to Pr. There is no application that would access the resulting fragments any differently.

ALGORITHM COM_MIN

Given: a relation R, the set of simple predicates Pr
Output: a $complete$ and $minimal$ set of simple predicates Pr' for Pr

$Rule$ 1: a relation or fragment is partitioned into at least two parts which are accessed differently by at least one application

- Initialization
  - find a $p_i \in Pr$ such that p_i partitions R according to Rule 1
  - set $Pr' = \{p_i\}$; $Pr \leftarrow Pr - \{p_i\}$; $F \leftarrow \{f_i\}$ where f_i is the fragment defined according to p_i.
- Interatively add predicates to Pr' until it is complete
  - find a $p_i \in Pr$ such that p_i partitions some f_h defined according to minterm predicate over Pr' according to Rule 1
  - set $Pr' = Pr' \cup \{p_i\}$; $Pr \leftarrow Pr - \{p_i\}$; $F \leftarrow F \cup \{f_i\}$
  - if $\exists p_k \in Pr'$ which is nonrelevant then
    $Pr' \leftarrow Pr' - \{p_k\}$; $F \leftarrow F - \{f_k\}$

PHORIZONTAL ALGORITHM

Employes COM_MIN to perform fragmentation

Given: a relation R, the set of simple predicates Pr
Output: a set of minterm predicates M according to which relation R is to be fragmented

- $Pr' \leftarrow$ COM_MIN(R, Pr)
- determine the set M of minterm predicates
- determine the set I of implications among $p_i \in Pr$
- eliminate the contradictory minterms from M

PHF – EXAMPLE

- two candidate relations: S and J
- fragmentation of S
  - application: check the salary info and determines raise
  - employee records kept at two sites, i.e. application runs at two sites
  - simple predicates
    - p_1: SAL $\leq$ 30000
    - p_2: SAL $>$ 30000
    - $Pr = \{p_1, p_2\}$ which is complete and minimal $Pr' = Pr$
  - minterm predicates
    - m_1: (SAL $\leq$ 30000) $\wedge$ (SAL $>$ 30000)
    - m_2: (SAL $\leq$ 30000) $\wedge$ NOT(SAL $>$ 30000)
    - m_3: NOT(SAL $\leq$ 30000) $\wedge$ (SAL $>$ 30000)
    - m_4: NOT(SAL $\leq$ 30000) $\wedge$ NOT(SAL $>$ 30000)

PHF – EXAMPLE (CONT)

- fragmentation of relation S (cont)
  - implications
    - i_1: (SAL $\leq$ 30000) $\Rightarrow$ NOT(SAL $>$ 30000)
    - i_2: NOT(SAL $\leq$ 30000) $\Rightarrow$ (SAL $>$ 30000)
    - i_3: (SAL $>$ 30000) $\Rightarrow$ NOT(SAL $\leq$ 30000)
    - i_4: NOT(SAL $>$ 30000) $\Rightarrow$ (SAL $\leq$ 30000)
- m_1 is contradictory to i_1, m_4 is contradictory to i_2

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000

PHF – EXAMPLE (CONT)

- fragmentation of relation J
  - applications:
    - find the name and budget of projects given their nos – issues at 4 sites
    - access project info according to budget – one site accesses $\leq$ 200000; other accesses $>$ 200000
  - simple predicates
    - application 1
      - p_1: LOC = "Montreal"
      - p_2: LOC = "New York"
      - p_3: LOC = "Paris"
      - p_4: LOC = "Boston"
    - application 2
      - p_5: BUDGET $\leq$ 200000
      - p_6: BUDGET $>$ 200000
    - $Pr = Pr' = \{p_1, p_2, p_3, p_4, p_5, p_6\}$

PHF – EXAMPLE (CONT)

- fragmentation of relation J (cont)
  - minterm fragments left after elimination
    - m_1: (LOC = “Montreal”) $\wedge$ (BUDGET $\leq$ 200000)
    - m_2: (LOC = “Montreal”) $\wedge$ (BUDGET $>$ 200000)
    - m_3: (LOC = “New York”) $\wedge$ (BUDGET $\leq$ 200000)
    - m_4: (LOC = “New York”) $\wedge$ (BUDGET $>$ 200000)
    - m_5: (LOC = “Paris”) $\wedge$ (BUDGET $\leq$ 200000)
    - m_6: (LOC = “Paris”) $\wedge$ (BUDGET $>$ 200000)
    - m_7: (LOC = “Boston”) $\wedge$ (BUDGET $\leq$ 200000)
    - m_8: (LOC = “Boston”) $\wedge$ (BUDGET $>$ 200000)

PHF – EXAMPLE (CONT)

JNO	JNAME	BUDGET	LOC
J1	Instrumentation	150000	Montreal

JNO	JNAME	BUDGET	LOC
J2	Database Develop.	135000	New York

JNO	JNAME	BUDGET	LOC
J3	CAD/CAM	250000	New York

JNO	JNAME	BUDGET	LOC
J4	Maintenance	310000	Paris

JNO	JNAME	BUDGET	LOC
J5	CAD/CAM	5000000	Boston

PHF – CORRECTNESS

- Completenness
  - since Pr' is complete and minimal, the selection predicates are complete
- Reconstruction
  - if relation R is fragmented into $F_R = \{R_1, \ldots, R_n\}$

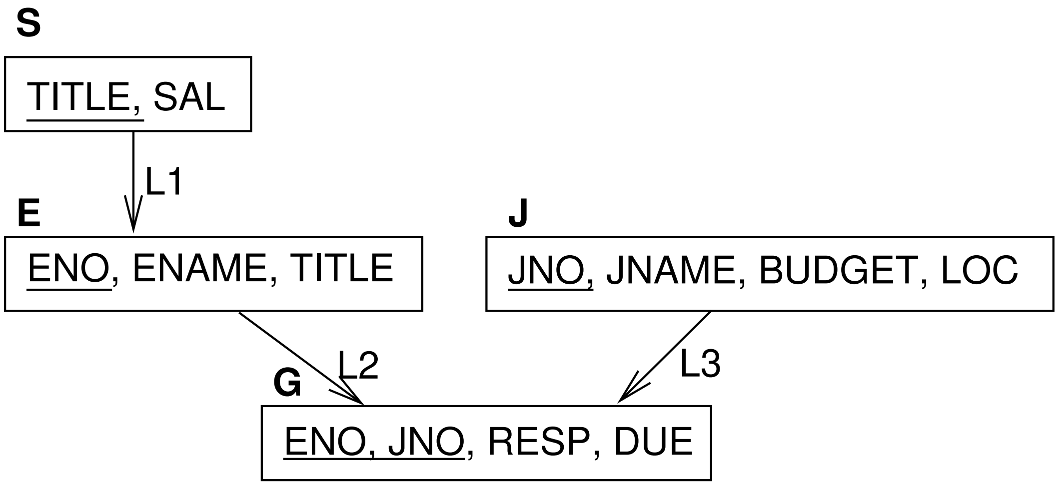
$$R = \cup_{\forall R_i \in F_R} R_i$$

- Disjointness
  - minterm predicates that form the basis of fragmentation should be mutually exclusive

DERIVED HORIZONTAL FRAGMENTATION (DHF)

defined on a number of relation of a link according to a selection operation specified on its owner

- each link is an equijoin
- equijoin can be implemented by means of semi-joins



- L1, L2 and L3 are links.
  owner(L1)=S
  member(L1)=E

DHF – DEFINITION

Given a link L where $owner(L) = S$ and $member(L) = R$, the derived horizontal fragments of R are defined as

$$R_i = R \ltimes S_i, \quad 1 \leq i \leq w$$

where w is the maximum number of fragments that will be defined on R and

$$S_i = \sigma_{F_i}(S)$$

where F_i is the formula according to which the primary horizontal fragments S_i is defined.

DHF – EXAMPLE

Given link L_1 where owner(L_1) = S and member(L_1) = E

$$E_1 = E \ltimes S_1, \quad E_2 = E \ltimes S_2$$

where

$$S_1 = \sigma_{SAL \leq 30000}(S), \quad S_2 = \sigma_{SAL > 30000}(S)$$

Fragment E1

ENO	ENAME	TITLE
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E7	R. Davis	Mech. Eng.

Fragment E2

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E8	J. Jones	Syst. Anal.

DHF – CORRECTNESS

- completeness
  - referential integrity
  - let R be the member relation of a link whose owner is relation S which is fragmented as $F_S = \{S_1, S_2, \ldots, S_n\}$. Furthermore, let A be the join attribute between R and S. Then, for each tuple r of R, there should be a tuple s of S such that

$$r[A] = s[A]$$

- reconstruction
  - same as primary horizontal fragmentation
- disjointness
  - simple join graphs between the owner and the member fragments

VERTICAL FRAGMENTATION

- has been studied within the centralized context
  - design methodology
  - physical clustering
- more difficult than horizontal because more alternatives exist
- two approaches:
  - grouping – attributes to fragments
  - splitting – relation to fragments

VERTICAL FRAGMENTATION

- overlapping fragments
  - grouping
- non-overlapping fragments
  - splitting
  - replicated key attributes are not considered as overlapping
  - easier to enforce functional dependencies (for integrity checking, etc.)

VF – INFORMATION REQUIREMENTS

- Application Information
  - attribute affinities
    - a measure that indicates how closely related the attributes are
    - obtained from more primitive usage data
  - attribute usage values
    - given a set of queries $Q = \{q_1, q_2, \ldots, q_p\}$ that will run on the relation $R[A_1, A_2, \ldots, A_n]$

$$use(q_i, A_j) = \begin{cases} 1 & \text{if } A_j \text{ is referenced by } q_i \\ 0 & \text{otherwise} \end{cases}$$

VF – DEFINITION OF $USE(q_i, A_j)$

Consider the following 4 queries for relation J

q_1:
```
SELECT  BUDGET
FROM    J
WHERE   JNO = Value
```

q_2:
```
SELECT  JNAME,BUDGET
FROM    J
```

q_3:
```
SELECT  JNAME
FROM    J
WHERE   LOC = VALUE
```

q_4:
```
SELECT  SUM(BUDGET)
FROM    J
WHERE   LOC = VALUE
```

Let A_1 =JNO, A_2 =JNAME, A_3 =BUDGET, A_4 =LOC
The attribute usage matrix USE is:

$$\begin{array}{cc} & \begin{array}{cccc} \text{A1} & \text{A2} & \text{A3} & \text{A4} \end{array} \\ \begin{array}{c} \text{q1} \\ \text{q2} \\ \text{q3} \\ \text{q4} \end{array} & \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

VF – AFFINITY MEASURE $aff(A_i, A_j)$

The attribute affinity measure between two attributes A_i and A_j of a relation $R[A_1, A_2, \ldots, A_n]$ wrt the set of applications $Q = (q_1, q_2, \ldots, q_p)$ is defined as:

$$aff(A_i, A_j) = \Sigma_{each\ query\ q_K\ that\ accesses\ A_i\ and\ A_j} \Sigma_{each\ site\ S_l} ref(S_l, q_k) \cdot acc_l(q_k)$$

where ref(S_l,q_k) is the number of accesses to both atttributes A_i and A_j for each execution of application q_k at site S_l, and $acc_l(q_k)$ is the access frequency with which the application q_k accesses data at site S_l over a period of time.

VF – AFFINITY MEASURE $aff(A_i, A_j)$

Assume each query in the previous example accesses the attributes once during each execution. Also assume the following access frequencies:

$$
\begin{array}{c}
 \\
q1 \\
q2 \\
q3 \\
q4
\end{array}
\begin{array}{c}
\begin{array}{ccc} S1 & S2 & S3 \end{array} \\
\begin{bmatrix}
15 & 20 & 10 \\
5 & 0 & 0 \\
25 & 25 & 25 \\
3 & 0 & 0
\end{bmatrix}
\end{array}
$$

Also assume ref(S_l,q_k) = 1 for all S_l and q_k.
Since only q_1 accesses both A_1 and A_3, we have

$$
\begin{aligned}
aff(A_1, A_3) &= 15 \times 1 + 20 \times 1 + 10 \times 1 \\
&= 45
\end{aligned}
$$

and the attribute affinity matrix AA is

$$
\begin{array}{c}
 \\
A1 \\
A2 \\
A3 \\
A4
\end{array}
\begin{array}{c}
\begin{array}{cccc} A1 & A2 & A3 & A4 \end{array} \\
\begin{bmatrix}
45 & 0 & 45 & 0 \\
0 & 80 & 5 & 75 \\
45 & 5 & 53 & 3 \\
0 & 75 & 3 & 78
\end{bmatrix}
\end{array}
$$

VF – CLUSTERING ALGO

- take the attribute affinity matrix AA and reorganize the attribute orders to form clusters where the attributes in each cluster demonstrate high affinity to one another
- Bond Energy Algorithm (BEA)
  - has been used for clustering of entities.
  - BEA finds an ordering of entities (in our case attributes) such that the global affinity measure

    $$AM = \sum_i \sum_j (aff\ of\ A_i\, and\, A_j\, with\, their\, neighbors)$$

    is maximized, where
    $aff\ of\ A_i\ and\ A_j\ with\ their\ neighbors$ =
    $aff(A_i, A_j) \cdot [aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})$ +
    $aff(A_{i-1}, A_j) + aff(A_{i+1}, A_j)]$
    and
    $aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) =$
    $aff(A_i, A_{n+1}) = 0$
  - Since the attribute affinity mattrix (AA) is symmetric, so
    $AM = \sum_i \sum_j aff(A_i, A_j) \cdot [aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})]$

BOND ENERGY ALGORITHM

Input: The AA matrix
Output: The clustered affinity matrix CA which is a perturbation of AA

- **Initialization:** Place and fix one of the columns of AA in CA
- **Iteration:**
  - Place the remaining $n - i$ columns in the remaining $i + 1$ positions in the CA matrix.
    - For each column, choose the placement that makes the most contribution to the global affinity measure
- **Row Order:** Order the rows according to the column ordering

BEA (CONT)

"Best" placement?

The net contribution to the global affinity measure of placing attribute A_k between attributes A_i and A_jk is

$$
\begin{aligned}
cont(A_i, A_k, A_j) \;=\; & 2 \cdot bond(A_i, A_k) + 2 \cdot bond(A_k, A_j) \\
- \; & 2 \cdot bond(A_i, A_j)
\end{aligned}
$$

where

$$bond(A_x, A_y) = \sum_{z=1}^{n} aff(A_z, A_x) \cdot aff(A_z, A_y)$$

BEA – EXAMPLE

Consider the following AA matrix and the corresponding CA matrix where A_1 and A_2 have been placed.

$$AA = \begin{array}{c} \\ A1 \\ A2 \\ A3 \\ A4 \end{array} \begin{array}{c} \begin{array}{cccc} A1 & A2 & A3 & A4 \end{array} \\ \begin{bmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{bmatrix} \end{array} \qquad CA = \begin{array}{c} \begin{array}{cccc} A1 & A2 & & \end{array} \\ \begin{bmatrix} 45 & 0 & & \\ 0 & 80 & & \\ 45 & 5 & & \\ 0 & 75 & & \end{bmatrix} \end{array}$$

Now we want to place A_3.
Ordering (0-3-1):
bond(A_0,A_1)=bond(A_0,A_3) = 0
bond(A_3,A_1)=45*45+5*0+53*45+3*0=4410

$$\begin{aligned} cont(A_0,\ A_3,\ A_1) &= 2 \cdot bond(A_0,\ A_3) + 2 \cdot bond(A_3,\ A_1) \\ &- \ 2 \cdot bond(A_0,\ A_1) \\ &= 2 \cdot 0 + 2 \cdot 4410 - 2 \cdot 0 = 8820 \end{aligned}$$

Ordering (1-3-2):

$$\begin{aligned} cont(A_1,\ A_3,\ A_2) &= 2 \cdot bond(A_1,\ A_3) + 2 \cdot bond(A_3,\ A_2) \\ &- \ 2 \cdot bond(A_l,\ A_2) \\ &= 2 \cdot 4410 + 2 \cdot 890 - 2 \cdot 225 = 10150 \end{aligned}$$

Ordering (2-3-4):
bond(A_2,A_3)=890, bond(A_3,A_4)=0, bond(A_2,A_4)=0
cont(A_2, A_3, A_4) = 1780

BEA – EXAMPLE (CONT)

Note that Ordering (1-3-2) has the maximun cont value, so we place attribute A_2 between A_1 and A_2. Therefore, the CA matrix has the form

$$\begin{array}{c c} & \begin{array}{ccc} A1 & A3 & A2 \end{array} \\ \begin{array}{c} A1 \\ A2 \\ A3 \\ A4 \end{array} & \begin{bmatrix} 45 & 45 & 0 \\ 0 & 5 & 80 \\ 45 & 53 & 5 \\ 0 & 3 & 75 \end{bmatrix} \end{array}$$

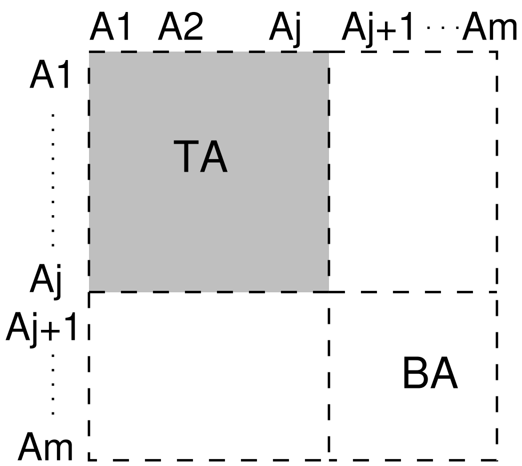
When A_4 is placed, the final form of the CA matrix (after rows are reorganized in the same order as the columns) is

$$\begin{array}{c c} & \begin{array}{cccc} A1 & A3 & A2 & A4 \end{array} \\ \begin{array}{c} A1 \\ A3 \\ A2 \\ A4 \end{array} & \begin{bmatrix} 45 & 45 & 0 & 0 \\ 45 & 53 & 5 & 3 \\ 0 & 5 & 80 & 75 \\ 0 & 3 & 75 & 78 \end{bmatrix} \end{array}$$

We see the creation of 2 clusters: $\{A_1, A_3\}$ and $\{A_2, A_4\}$. This clustering indicates how the attributes of relation J should be split. When CA is big, usually more than two clusters are formed and there are more than one candidate partitionings.

VF – ALGORITHM

Consider the below clustered attribute matrix.
How can a set of clustered attributes $\{A_1, A_2, \ldots, A_n\}$ be divided into two (or more) sets $\{A_1, A_2, \ldots, A_j\}$ and $\{A_{j+1}, \ldots, A_n\}$ such that there are no (or minimal) applications that access both (or more than one) of the sets?



VF – ALGORITHM (CONT)

Define

TQ =	set of applications that access only TA
BQ =	set of applications that access only BA
OQ =	set of applications that access both

and

CTQ = total number of accesses to attributes by applications that access only TA

$$CTQ = \sum_{q_i \in TQ} \sum_{\forall S_j} ref(S_j, q_i) \cdot acc_j(q_i)$$

CBQ = total number of accesses to attributes by applications that access only BA

$$CBQ = \sum_{q_i \in BQ} \sum_{\forall S_j} ref(S_j, q_i) \cdot acc_j(q_i)$$

COQ = total number of accesses to attributes by applications that access both

$$COQ = \sum_{q_i \in OQ} \sum_{\forall S_j} ref(S_j, q_i) \cdot acc_j(q_i)$$

Then find the point along the diagonal that maximizes

$$CTQ \cdot CBQ - COQ^2$$

VF – ALGORITHM (CONT)

Two problems

- cluster forming in the middle of the CA matrix
  - shift a row up, and a column left, and apply the algorithm to find the “best” partitioning point
  - do this for all possible shifts
  - cost $O(n^2)$ (n attributes)
- more than two clusters
  - m-way partitioning
  - try 1, 2, $\ldots$, $m-1$ split points along diagonal and try to find the best point for each of these
  - cost $O(2^m)$

Example:
When the algorithm is applied to the CA matrix obtained in the previous example, the result is the definition of fragments F_J = $\{J_1,J_2\}$, where J_1=$\{A_1,A_3\}$ and J_2=$\{A_1,A_2,A_4\}$ since A_1 is the key of J. Thus

J_1 = {JNO, BUDGET}
J_2 = {JNO, JNAME, LOC}

VF – CORRECTNESS

a relation R, defined over attribute set A and key K, generates the vertical partitioning $F_R = \{R_1, R_2, \ldots, R_r\}$

- completeness
  - the following should be true for A:

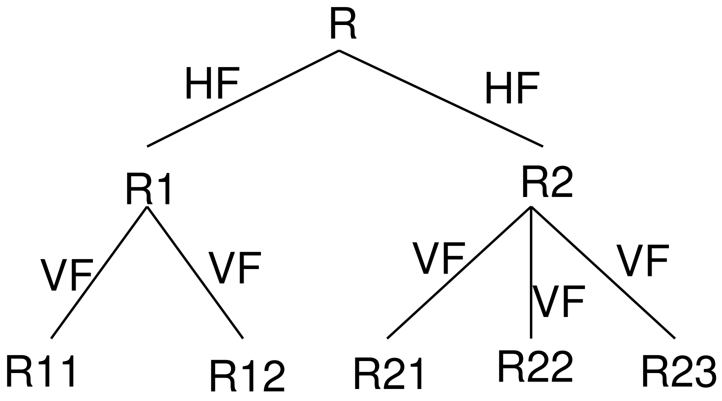
$$A = \cup A_{R_i}$$

- reconstruction
  - reconstruction can be achieved by

$$R = \bowtie_K R_i \quad \forall R_i \in F_R$$

- disjointness
  - TID's are not considered to be overlapping since they are maintained by the system
  - duplicated keys are not considered to be overlapping

HYBRID FRAGMENTATION



FRAGMENT ALLOCATION

- Problem Statement
  - Given

$$
\begin{aligned}
F = &\quad \{F_1,\ F_2,\ \ldots,\ F_n\} \text{ (fragments)}\\
S = &\quad \{S_1,\ S_2,\ \ldots,\ S_n\} \text{ (network sites)}\\
Q = &\quad \{q_1,\ q_2,\ \ldots,\ q_n\} \text{ (applications)}
\end{aligned}
$$

  Find the "optimal" distribution of F to S

- Optimality
  - minimum cost
    - communication + storage + processing (read and update)
    - cost in terms of time (usually)
  - performance – response time and/or throughput
  - constraints – per site constraints (storage and processing)

INFORMATION REQUIREMENTS

- **Database information**
  - selectivity of fragments
  - size of a fragment
- **Application information**
  - access types (update or read) and numbers
  - access localities
- **Communication network information**
  - unit cost of storing data at a site
  - unit cost of processing at a site
- **Computer system information**
  - bandwidth
  - latency
  - communication overhead

ALLOCATION

File Allocation (FAP) vs Database Allocation (DAP):

- fragments are not individual files
  - relationships have to be maintained
- access to databases is more complicated
  - remote file access model not applicable
  - relationship between allocation and query processing
- cost of integrity enforcement should be considered
- cost of concurrency control should be considered

ALLOCATION - INFORMATION REQUIREMENTS

- **Database information**
  - selectivity of fragments
  - size of a fragment
- **Application information**
  - number of read accesses of a query to a fragment
  - number of update accesses of a query to a fragment
  - a matrix indicating which queries update which fragment
  - a similar matrix for retrievals
  - originating site of each query
- **Site information**
  - unit cost of storing data at a site
  - unit cost of processing at a site
- **Network information**
  - communication cost/frame between two sites
  - frame size

ALLOCATION MODEL

General Form

min(Total Cost)

subject to

response time constraint

storage constraint

processing constraint

Decision Variable

$$x_{ij} = \begin{cases} 1 & \text{if fragment } F_i \text{ is stored at site } S_j \\ 0 & \text{otherwise} \end{cases}$$

ALLOCATION MODEL (CONT)

- Total Cost

$$\sum_{all\ queries} query\ processing\ cost\ +$$
$$\sum_{all\ sites}\ \sum_{all\ fragments} cost\ of\ storing\ a\ fragment\ at\ a\ site$$

- Storage Cost (of fragment F_j at S_k) - STC_jk
  - (unit storage cost at S_k) $\cdot$ (size of F_j) $\cdot$ x_{jk}
- Query Processing Cost (for one query) - QPC_i
  - processing component + transmission component

ALLOCATION MODEL (CONT)

- Query Processing Cost - QPC_i

  (1) processing component

  access $+$ integrity enforcement $+$ concurrency control

  – access cost - AC_i

    $\sum_{all\ sites\ S_k} \sum_{all\ fragments\ F_j}$ (no. of updates + no. of reads) $\times\ x_{jk} \times$ (local processing cost at a site)

    i.e.

    AC_i = $\Sigma\forall S_k\ \ \Sigma\forall F_j\ (u_{ij}$ * UR_{ij} + r_{ij} * RR_{ij}) * x_{jk} * $LPC_k)$

    Note that (UR_{ij} + RR_{ij}) gives the total number of update and retrieval accesses. LPC_k is the cost of processing one unit of work at site S_k.

    u_{ij} (or r_{ij}) = 1 if query q_i updates (or retrieval) the segment F_j, else is 0.

  – integrity enforcement and concurrency control costs can be similarly calculated

ALLOCATION MODEL (CONT)

- Query Processing Cost

  (2) Transmission component

  cost of processing updates $+$ cost of processing retrievals

  – cost of updates

  $$\sum_{all\ sites} \sum_{all\ fragments} \text{update message cost} +$$
  $$\sum_{all\ sites} \sum_{all\ fragments} \text{acknowledgement cost}$$

  Note: All fragment replicas need to be updated.

  – retrieval cost

  $$\sum_{all\ fragments} \min_{all\ sites} (\text{cost of retrieval command} + \text{cost of sending back the result})$$

  Note: Only the site that yields the minimum total transmission cost should be selected for the execution of the operation.

ALLOCATION MODEL (CONT)

- Constraints
  - response time

    execution time of query

    $\leq$ max. allowable response time for that query
  - storage constraint (for a site)

    $\Sigma_{all\ fragments}$ storage requirement of a fragment

    at that site $\leq$ storage capacity at that site
  - processing constraint (for a site)

    $\Sigma_{all\ queries}$ query processing load at that site

    $\leq$ processing capacity of that site

ALLOCATION MODEL (CONT)

- Solution methods
  - File allocation (FAP) is NP-complete
  - Database allocation (DAP) also NP-complete
- Heurstics based on
  - single commodity warehouse location (for FAP)
  - knapsack problem
  - branch and bound techniques
  - network flow

ALLOCATION MODEL (CONT)

- attempts to reduce the solution space
  - assume all candidate partitionings known; select the "best" partitioning
  - ignore replication at first

PROBLEM – SEPARATION OF DESIGN STEPS