Distributed Database design: fragmentation & allocation

Material from:
Principles of Distributed Database Systems
Özsu, M. Tamer, **Valduriez**, Patrick, 3rd ed. 2011

+ slides from H. Garcia Molina.

Presented by C. Roncancio

Distribution Design

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

Distribution Design Issues

❶ Why fragment at all?

❷ How to fragment?

❸ How much to fragment?

❹ How to test correctness?

❺ How to allocate?

❻ Information requirements?

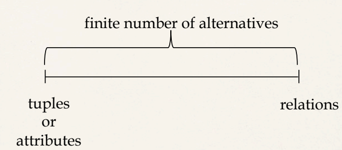
Fragmentation

- Can't we just distribute relations?
- What is a reasonable unit of distribution?
  - relation
    - views are subsets of relations ➔locality
    - extra communication
  - 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 (especially integrity enforcement) more difficult

Fragmentation

- Horizontal Fragmentation (HF)
  - Primary Horizontal Fragmentation (PHF)
  - Derived Horizontal Fragmentation (DHF)
- Vertical Fragmentation (VF)
- Hybrid Fragmentation (HF)

Degree of Fragmentation



Finding the suitable level of partitioning within this range

Fragmentation Alternatives – Horizontal

PROJ$_1$: projects with budgets less than $200,000

PROJ$_2$: projects with budgets greater than or equal to $200,000

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

Fragmentation Alternatives – Horizontal

PROJ$_1$: projects with budgets less than $200,000

PROJ$_2$: projects with budgets greater than or equal to $200,000

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

PROJ$_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York

PROJ$_2$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

Correctness of Fragmentation

- Completeness
  - Decomposition of relation R into fragments $R_1, R_2, ..., R_n$ is complete if and only if each data item in R can also be found in some R_i
- Reconstruction
  - If relation R is decomposed into fragments $R_1, R_2, ..., R_n$, then there should exist some relational operator ∇ such that

    $R = \nabla_{1 \le i \le n} R_i$
- Disjointness
  - If relation R is decomposed into fragments $R_1, R_2, ..., R_n$, and 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:

  If $\frac{\text{read-only queries}}{\text{update queries}} \ll 1$, replication is advantageous,

  otherwise replication may cause problems

Comparison of Replication Alternatives

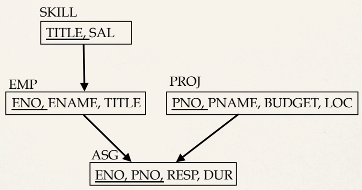
	Full-replication	Partial-replication	Partitioning
QUERY PROCESSING	Easy	← Same Difficulty →	
DIRECTORY MANAGEMENT	Easy or Non-existant	← Same Difficulty →	
CONCURRENCY CONTROL	Moderate	Difficult	Easy
RELIABILITY	Very high	High	Low

Information Requirements

- Four categories:
  - Database information
  - Application information
  - Communication network information
  - Computer system information

PHF – Information Requirements

- Database Information
  - relationship



  - cardinality of each relation: *card*(*R*)

PHF - Information Requirements

- 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 we define $Pr = \{p_1, p_2, \ldots, p_m\}$

    Example :

    PNAME = "Maintenance"

    BUDGET ≤ 200000
  - **minterm predicates** : Given R and $Pr = \{p_1, p_2, \ldots, p_m\}$

    define $M = \{m_1, m_2, \ldots, m_r\}$ as

    $M = \{ m_i \mid m_i = \bigwedge_{p_j \in Pr} p_j^* \}, 1 \leq j \leq m, 1 \leq i \leq z$

    where $p_j^* = p_j$ or $p_j^* = \neg(p_j)$.

PHF – Information Requirements

Example

m_1: PNAME="Maintenance" ∧ BUDGET≤200000

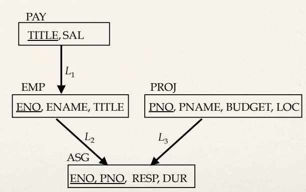
m_2: **NOT**(PNAME="Maintenance") ∧ BUDGET≤200000

m_3: PNAME= "Maintenance" ∧ **NOT**(BUDGET≤200000)

m_4: **NOT**(PNAME="Maintenance") ∧ **NOT**(BUDGET≤200000)

Derived Horizontal Fragmentation

- Defined on a member 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 semijoins.



DHF – Definition

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

$$R_i = R \ltimes_F S_i, 1 \leq i \leq w$$

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 fragment S_i is defined.

Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Example

$Pay_1 = \sigma_{SAL \leq 30000}(Pay)$
$Pay_2 = \sigma_{SAL > 30000}(Pay)$

PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

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

DHF – Example

Given link L_1 where owner(L_1)=Pay and member(L_1)=EMP

$EMP_1 = EMP \ltimes Pay_1$
$EMP_2 = EMP \ltimes Pay_2$

Where

$Pay_1 = \sigma_{SAL \leq 30000}(Pay)$
$Pay_2 = \sigma_{SAL > 30000}(Pay)$

EMP_1

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

EMP_2

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 $F_S = \{S_1, S_2, ..., S_n\}$. Furthermore, let A be the join attribute between R and S. Then, for each tuple t of R, there should be a tuple t' of S such that

    $t[A] = t'[A]$
- 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

Fragmentation Alternatives – Vertical

$PROJ_1$: information about project budgets

$PROJ_2$: information about project names and locations

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

Fragmentation Alternatives – Vertical

$PROJ_1$: information about project budgets

$PROJ_2$: information about project names and locations

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

$PROJ_1$

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000
P5	500000

$PROJ_2$

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Develop.	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris
P5	CAD/CAM	Boston

Vertical Fragmentation

- Overlapping fragments
  - grouping
- Non-overlapping fragments
  - splitting

We do not consider the replicated key attributes to be overlapping.

Advantage:

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
    - This is obtained from more primitive usage data
  - Attribute usage values
    - Given a set of queries $Q = \{q_1, q_2,\ldots, q_q\}$ that will run on the relation $R[A_1, A_2,\ldots, A_n]$,

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

$use(q_i,\bullet)$ can be defined accordingly

VF – Definition of $use(q_i,A_j)$

Consider the following 4 queries for relation PROJ

q_1: **SELECT** BUDGET
FROM PROJ
WHERE PNO=Value

q_2: **SELECT** PNAME,BUDGET
FROM PROJ

q_3: **SELECT** PNAME
FROM PROJ
WHERE LOC=Value

q_4: **SELECT** **SUM**(BUDGET)
FROM PROJ
WHERE LOC=Value

Let A_1= PNO, A_2= PNAME, A_3= BUDGET, A_4= LOC

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

VF – Affinity Measure $aff(A_i,A_j)$

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

$$aff(A_i, A_j) = \sum_{\text{all queries that access } A_i \text{ and } A_j} \text{(query access)}$$

$$\text{query access} = \sum_{\text{all sites}} \text{access frequency of a query} * \frac{\text{access}}{\text{execution}}$$

VF – Calculation of $aff(A_i, A_j)$

Assume each query in the previous example accesses the attributes once during each execution.

Also assume the access frequencies →

	S_1	S_2	S_3
q_1	15	20	10
q_2	5	0	0
q_3	25	25	25
q_4	3	0	0

Then

$aff(A_1, A_3)$ = 15*1 + 20*1+10*1
= 45

and the attribute affinity matrix AA is →

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

VF – Calculation of $aff(A_i, A_j)$

Assume each query in the previous example accesses the attributes once during each execution.

Also assume the access frequencies →

	S_1	S_2	S_3
q_1	15	20	10
q_2	5	0	0
q_3	25	25	25
q_4	3	0	0

Then

$aff(A_1, A_3)$ = 15*1 + 20*1+10*1
= 45

and the attribute affinity matrix AA is →

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

VF – Clustering Algorithm

- 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 is maximized.

$$AM = \sum_i \sum_j \text{(affinity of } A_i \text{ and } A_j \text{ with their neighbors)}$$

VF – Correctness

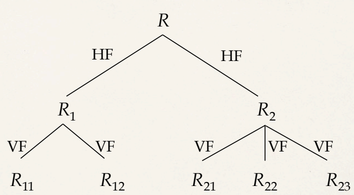
A relation R, defined over attribute set A and key K, generates the vertical partitioning $F_R = \{R_1, R_2, \ldots, R_r\}$.

- Completeness
  - The following should be true for A:

    $A = \bigcup A_{R_i}$
- Reconstruction
  - Reconstruction can be achieved by

    $R = \bowtie_K R_i, \forall R_i \in F_R$
- 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

  $F = \{F_1, F_2, \ldots, F_n\}$ fragments

  $S = \{S_1, S_2, \ldots, S_m\}$ network sites

  $Q = \{q_1, q_2, \ldots, q_q\}$ applications

  Find the "optimal" distribution of F to S.
- Optimality
  - Minimal cost
    - Communication + storage + processing (read & update)
    - Cost in terms of time (usually)
  - Performance

    Response time and/or throughput
  - Constraints
    - Per site constraints (storage & processing)

Information Requirements

- Database information
  - selectivity of fragments
  - size of a fragment
- Application information
  - access types 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 query to a fragment
  - A matrix indicating which queries update which fragments
  - 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

- Total Cost

$$\sum_{\text{all queries}} \text{query processing cost} +$$

$$\sum_{\text{all sites}} \sum_{\text{all fragments}} \text{cost of storing a fragment at a site}$$

- Storage Cost (of fragment F_j at S_k)

(unit storage cost at S_k) * (size of F_j) * x_{jk}

- Query Processing Cost (for one query)

processing component + transmission component

Allocation Model

- Query Processing Cost

Processing component

access cost + integrity enforcement cost + concurrency control cost

- Access cost

$$\sum_{\text{all sites}} \sum_{\text{all fragments}} (\text{no. of update accesses+ no. of read accesses}) * x_{ij} * \text{local processing cost at a site}$$

- Integrity enforcement and concurrency control costs
  - Can be similarly calculated

Allocation Model

- Query Processing Cost

Transmission component

cost of processing updates + cost of processing retrievals

- Cost of updates

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

- Retrieval Cost

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

Allocation Model

- Constraints
  - Response Time

    execution time of query ≤ max. allowable response time for that query
  - Storage Constraint (for a site)

$$\sum_{\text{all fragments}} \text{storage requirement of a fragment at that site} \leq \text{storage capacity at that site}$$

  - Processing constraint (for a site)

$$\sum_{\text{all queries}} \text{processing load of a query at that site} \leq \text{processing capacity of that site}$$

Allocation Model

- Solution Methods
  - FAP is NP-complete
  - DAP also NP-complete
- Heuristics based on
  - single commodity warehouse location (for FAP)
  - knapsack problem
  - branch and bound techniques
  - network flow

Allocation Model

- Attempts to reduce the solution space
  - assume all candidate partitionings known; select the "best" partitioning
  - ignore replication at first
  - sliding window on fragments

Fragmentation / sharding

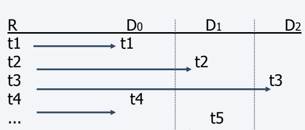
- A partition forms a « shard »
- Fragmentation based on FAQ or known access patterns
- Automatic partitioning, sharding?
- Sharding is often related to shared nothing architectures

Three common horizontal partitioning techniques

- Round robin
- Hash partitioning
- Range partitioning

From H. Garcia Molina

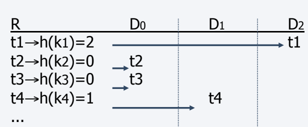
• Round robin



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

From H. Garcia Molina

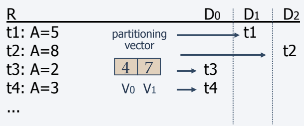
• Hash partitioning



- 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

From H. Garcia Molina

- Range partitioning



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

From H. Garcia Molina

Conclusion

- Fragmentation
  - Decomposition / reconstruction
  - Queries & integrity constraints
  - Horizontal simple / derived , vertical, hybrid
  - Properties: reconstruction, completeness, disjunction
- *Sharding*
- Allocation / duplication
  - Cost model
  - Iterative approach

80

Distributed DBMS © M. T. Özsu & P. Valduriez Ch.3/80