Distributed Database Design (Chapter 5)

- Top-Down Approach: The database system is being designed from scratch.
  - Issues: fragmentation & allocation
- Bottom-up Approach: Integration of existing databases (Chapter 15)
  - Issues: Design of the export and global schemas.

Design Consideration (1)

The organization of distributed systems can be investigated along three dimensions:

Level of sharing

1. No sharing: Each application and its data execute at one site; no communication with other program or access to any data file at other sites
2. Data sharing: Programs are replicated at all sites, but data files are not

   Data is moved to where the query is originated
3. Data + Program Sharing: Both data and programs may be shared.

   A program at one site can request for a service from another program at another site

Design Consideration (2)

Assess Pattern

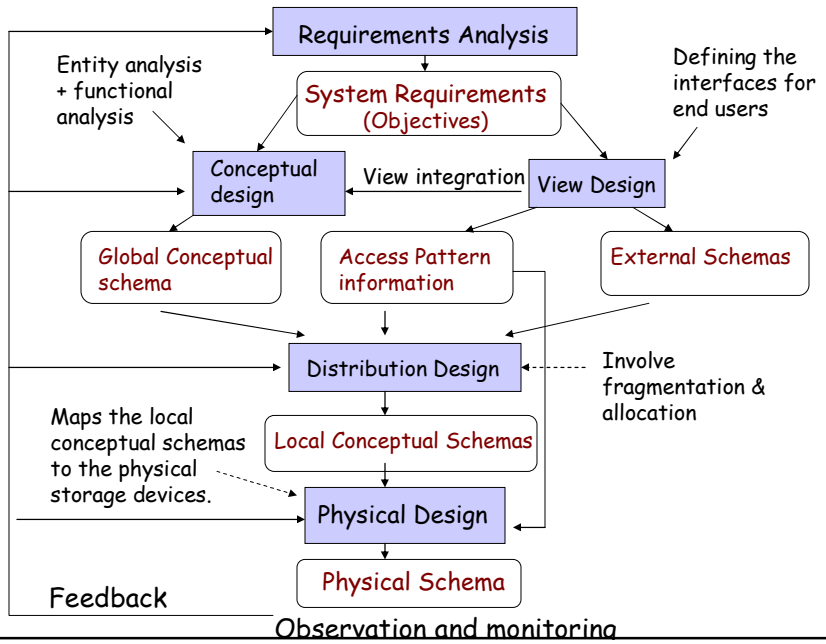
1. Static: Access patterns do not change.
2. Dynamic: Access patterns change over time.

   How dynamic the access pattern is?

Level of Knowledge

1. No information: Designers do not have the knowledge of the access pattern at all
2. Partial information: Access patterns may deviate from the predictions.
3. Complete information: Access patterns can reasonably be predicted and are not much different from the predictions

TOP-DOWN DESIGN PROCESS



Requirement Analysis

- Environment of the system
- Performance, reliability, availability, expandability, and cost.

View Design

Deal with defining the end user interfaces.

Conceptual Design

Consist of

- Entity Analysis
  - Determine entities and relationships among them.
- Functional Analysis

- Conceptual design can be seen as an integration of the user views.
- View integration is used to ensure that the conceptual model support both existing and future applications.

Functional Analysis

- Process of understanding and documenting basic business activities with which the organization is concerned.
- Function is triggered by events and can be defined as tasks that must be carried out as a direct result of an event.
- Output contains
  - Frequency of use of the function
  - How each attributed is used in the function.
  - Requirement on response time, availability, how up-to-date data should be.

Design Issues

- Why fragment at all?
- How should and how much should we fragment?
- A way to test correctness of the fragmentation?
- How to allocate fragments?
- Necessary information for fragmentation and allocation?

Why fragment at all?

- Reasons:
  - Interquery concurrency
    - Several queries can be executed in parallel.
  - Intraquery concurrency
    - Allowing parallel execution of a single query.
- Disadvantages:
  - Vertical fragmentation may incur overhead.
  - Attributes participating in a dependency may be allocated to different sites.
    - Integrity checking is more costly.
  - If there exists non disjoint fragments → more overhead

Fragmentation Alternatives (M)

J

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal
J2	Database Dev.	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	350,000	Orlando

Horizontal Partitioning

JP1

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal
J2	Database Dev.	135,000	New York

JP2

JNO	JNAME	BUDGET	LOC
J3	CAD/CAM	250,000	New Yorkl
J4	Maintenance.	310,000	Orlando

Vertical Partitioning

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

JNO	JNAME	LOC
J1	Instrumentation	Montreal
J2	Database Devl	New York
J3	CAD/CAM	New York
J4	Maintenance	Orlando

Degree of Fragmentation

- Application views are usually subsets of relations. Hence, it is only natural to consider subsets of relations as distribution units.
  - No fragment → every tuple is a fragment
  - No fragment →every attribute is a fragment
- The appropriate degree of fragmentation is dependent on the applications.

Correctness Rules

- Vertical Partitioning
  - Lossless decomposition
  - Dependency preservation
  - Disjointness on the nonprimary key attributes
- Horizontal Partitioning
  - Disjoint fragments

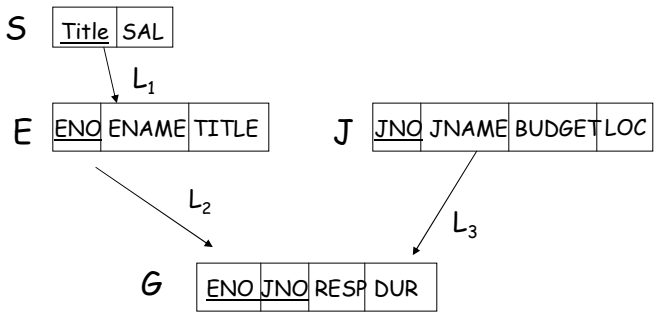
Allocation Alternatives

- Partitioning: No replication
- Partial Replication: Some fragments are replicated.
- Full Replication: Database exists in its entirety at each site.

Information Requirements for Horizontal Fragmentation

- Database Information
  - Global conceptual schema
  - The relationship between relations
- Application Information

Database Information Notations



→:1-to-many relationship
Owner(L1) = S = Source relation
Member(L1) = E = Target relation
Direct Link means equi-join

Application Information

- Qualitative Information
  - The fundamental qualificative information consists of the predicates used in user queries.
  - Analyze user queries based on 80/20 rule: 20% of user queries account for 80% of the total data access.
    - ☞ One should investigate the more important queries.
- Quantitative Information
  - Minterm Selectivity sel(m_i): number of tuples that would be accessed by a query specified according to a given minterm predicate.
  - Access Frequency acc(m_i): the access frequency of a given minterm predicate in a given period.

Qualitative information guides the fragmentation activity.
Quantitative information guides the allocation activity.

Simple Predicates

Given a relation R(A_1, A_2, …, A_n) where A_i has domain D_i, a simple predicate p_j defined on R has the form

p_j: $A_i \ \theta \ Value$
where $\theta \in \{=, <, \neq, \leq, >, \geq\}$ and $Value \in D_i$

Example:

J

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal
J2	Database Dev.	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	350,000	Orlando

Simple predicates:

p_1: JNAME= "Maintenance"
P_2: BUDGET<= 200,000

MINTERM PREDICATE

Given a set of simple predicates for relation R.

$P=\{p_1, p_2, \ldots, p_m\}$

The set of minterm predicates

$M=\{m_1, m_2, \ldots, m_n\}$

is defined as

$M=\{m_i \mid m_i = \bigwedge_{P_j \in P} P_j^* \}$

where $P_j^* = p_j$ or $P_j^* = \neg p_j$

TITLE	SAL
Elect. Eng.	40,000
Syst. Analy.	54,000
Mech. Eng.	32,000
Programmer	42,000

Minterm predicate is a conjunction of simple predicates

Possible simple predicates:

P_1: TITLE="Elect. Eng."
P_2: TITLE="Syst. Analy"
P_3: TITLE="Mech. Eng."
P_4: TITLE="Programmer"
P_5: SAL<=35,000
P_6: SAL > 35,000

Some corresponding minterm predicates:

$m_1 : TITLE = "Elect.Eng." \wedge SAL \leq 35{,}000$

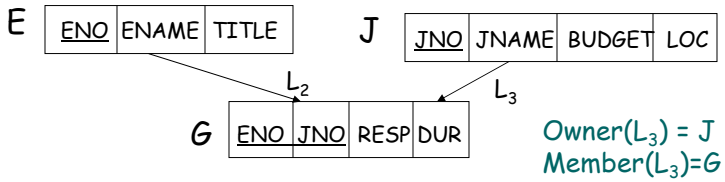
$m_2 : TITLE \neq "Elect.Eng" \wedge SAL > 35{,}000$

Horizontal Fragmentation

- Primary Horizontal Fragmentation
- Derived Horizontal Fragmentation

Primary Horizontal Fragmentation

A primary horizontal fragmentation of a relation is defined by a selection operation on the relation



A possible fragmentation of J is defined as follows:

$$JP1 = \sigma_{BUDGET \leq 200,000}(J)$$
$$JP2 = \sigma_{BUDGET > 200,000}(J)$$

Horizontal Fragments

Thus, a horizontal fragment R_i of relation R consists of all the tuples of R that satisfy a minterm predicate m_i.

There are as many horizontal fragments (also called minterm fragments) as there are minterm predicates.

Which minterm predicates should we use?

We have to decide on the set of simple predicates that are the basis for the minterm predicates.

Desirable properties of the set of simple predicates

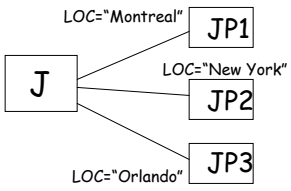
The set should be complete and minimal.

Informally, the set should include only predicates with attributes and conditions that are used in the applications

Completeness (1)

A set of simple predicate Pr is said to be complete if and only if there is an equal probability of access by every application to any tuple belonging to any minterm fragment that is defined according to Pr.

$$JP_1 = \sigma_{LOC = "MONTREAL"}(J)$$
$$JP_2 = \sigma_{LOC = "NewYork"}(J)$$
$$JP_3 = \sigma_{LOC = "Orlando"}(J)$$



Case 1: The only application that accesses J wants to access the tuples according to the location (any location).

The set of simple predicates

Pr= { LOC="Montreal", LOC="New York", LOC="Orlando" }

is complete because each tuple of each fragment has the same probability of being accessed.

Completeness (2)

Example:

JP1

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal

JP2

JNO	JNAME	BUDGET	LOC
J2	GUI	135,000	New York
J3	CAD/CAM	250,000	New York

JP3

JNO	JNAME	BUDGET	LOC
J4	Database Dev.	310,000	Orlando

Note: Completeness is a desirable property because a complete set defines fragments that are not only logically uniform in that they all satisfy the minterm predicate, but statistically homogeneous.

Case 2: There is a second application which accesses only those project tuples where the budget is less than $200,000.

The previous Pr is not complete since some tuple in JPi has higher access probability

- For example, tuple "J2" has higher access probability than tuple "J3" in JP2 (2 applications access J2, but one application access J3)
- To make the set complete, we need to add (BUDGET<= 200,000, BUDGET>200,000) to Pr.

Example

The only application that accesses J wants to access projects located in New York.

- The set of simple predicates

  Pr= {LOC="Montreal", LOC="New York", Loc!="New York", LOC="Orlando"}

Is this set Pr complete?

The only application that accesses J wants to access project located in New York.

JP1

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal

JP2

JNO	JNAME	BUDGET	LOC
J2	GUI	135,000	New York
J3	CAD/CAM	250,000	New York

JP3

JNO	JNAME	BUDGET	LOC
J4	Database Dev.	310,000	Orlando

- Is Loc="Orlando" relevant? No
- Is Loc="New York" relevant? Yes
- Is Loc="Montreal" relevant? No

Minimality

Minimal

If all the predicates of a set Pr are relevant, Pr is minimal.

That is, there should be at least one application that accesses fragments f_i and f_j differently.

i.e., The simple predicate p_i should be relevant in determining a fragmentation.

Relevant

Let m_i and m_j 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_j contains $\neg p_i$.

Also, let f_i and f_j be two fragments defined according to m_i and m_j, respectively. Then p_i is relevant if and only if

Access frequency

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

- Card(fi): Number of tuples in fi

A Complete and Minimal Example

Two applications:

1. One application accesses the tuples according to the location.
2. Another application accesses only those project tuples where the budget is at most $200,000.

Case 1: Pr={Loc="Montreal", Loc="New York", Loc="Orlando", BUDGET<=200,000,BUDGET>200,000}
is complete and minimal?

Case 2: If, however, we were to add the predicate JNAME= "Instrumentation" to Pr, the resulting set would not be minimal since the new predicate is not relevant with respect to the applications.

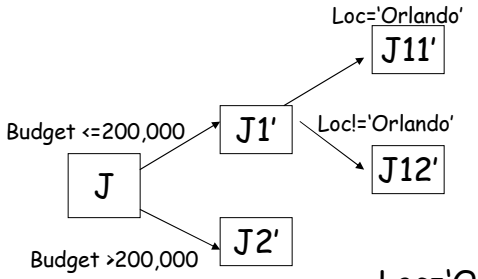
(A)

Case 1: Pr={Loc="Montreal", Loc="New York", Loc="Orlando", BUDGET<=200,000,BUDGET>200,000}

J

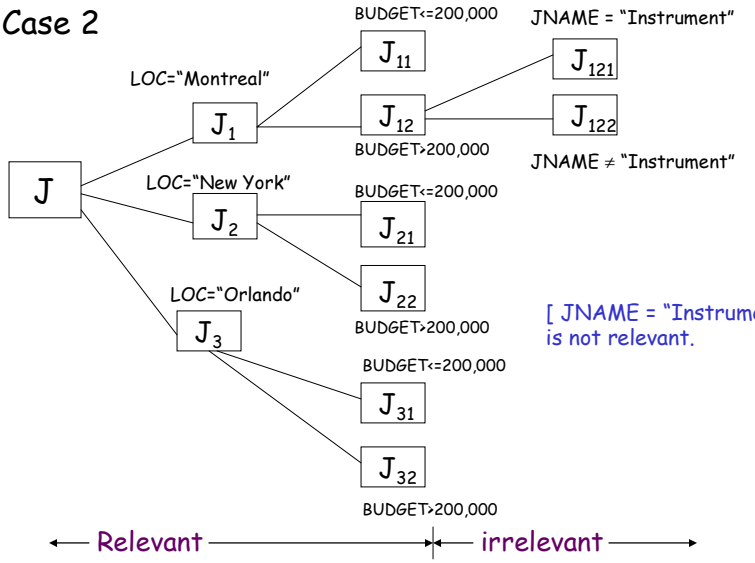
JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal
J2	Database Dev.	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	350,000	Orlando

Pr is complete from the previous example. Is Pr minimal?



Loc='Orlando' is relevant.

Case 2



[JNAME = "Instrument"] is not relevant.

Determine the set of meaningful minterm predicates

Application: Take the salary and determine a raise accordingly.

Fragmentation: The employee records are managed in two places, one handling the records of those with salary less than or equal to $30,000 and the other handling the records of those who earn more than $30,000.

Pr={SAL<=30,000,SAL>30,000} is complete and minimal.

The minterm predicates:

$m_1 : (SAL \le 30{,}000) \wedge (SAL > 30{,}000)$
$m_2 : (SAL \le 30{,}000) \wedge \neg(SAL > 30{,}000)$
$m_3 : \neg(SAL \le 30{,}000) \wedge (SAL > 30{,}000)$
$m_4 : \neg(SAL \le 30{,}000) \wedge \neg(SAL > 30{,}000)$

Implications:

$i_1 : (SAL \le 30{,}000) \Rightarrow \neg(SAL > 30{,}000)$
$i_2 : \neg(SAL \le 30{,}000) \Rightarrow (SAL > 30{,}000)$
$i_3 : (SAL > 30{,}000) \Rightarrow \neg(SAL \le 30{,}000)$
$i_4 : \neg(SAL > 30{,}000) \Rightarrow (SAL \le 30{,}000)$

$i_1 \Rightarrow m_1$ is contradictory

$i_2 \Rightarrow m_4$ is contradictory

Therefore, we are left with M = {m_2, m_3}

Invalid Implications

J

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal
J2	Database Dev.	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	350,000	Orlando

Simple predicates

p_1: LOC="Montreal"
p_2: LOC="New York"
p_3: LOC="Orlando"
p_4: BUDGET<=200,000
p_5: BUDGET>200,000

VALID Implications

$i_1 : p_1 \Rightarrow \neg p_2 \wedge \neg p_3$
$i_2 : p_2 \Rightarrow \neg p_1 \wedge \neg p_3$
$i_3 : p_3 \Rightarrow \neg p_1 \wedge \neg p_2$
$i_4 : p_4 \Rightarrow \neg p_5$
$i_5 : p_5 \Rightarrow \neg p_4$
$i_6 : \neg p_4 \Rightarrow p_5$
$i_7 : \neg p_5 \Rightarrow p_4$

INVALID Implications

$i_8 : LOC = "Montreal" \Rightarrow \neg(BUDGET > 200{,}000)$
$i_9 : LOC = "Orlando" \Rightarrow \neg(BUDGET \le 200{,}000)$

Implications should be defined according to the semantics of the database, not according to the current values.

Algorithms

Rule 1: fundamental rule of *completeness and minimality*, which states that a relation or fragment is partitioned "into at least two parts which are accessed differently by at least one application.

f_i of Pr': fragment f_i defined according to a minterm predicate defined over the simple predicates of Pr'.

Algorithm 5.1 *COM_MIN*
input: R: relation; Pr: set of simple predicates
output: Pr': set of simple predicates
declare
F: set of minterm fragments
begin
find a $p_i \in Pr$ such that p_i partitions R according to *Rule* 1
$Pr' \leftarrow p_i$
$Pr \leftarrow Pr - p_i$
$F \leftarrow f_i$ $\{f_i$ is the minterm fragment according to $p_i\}$
do
begin
find a $p_j \in Pr$ such that p_j partitions some f_k of Pr' according to *Rule* 1
$Pr' \leftarrow Pr' \cup p_j$
$Pr \leftarrow Pr - p_j$
$F \leftarrow F \cup f_j$
if $\exists p_k \in Pr'$ which is nonrelevant **then**
begin
$Pr' \leftarrow Pr' - p_k$
$F \leftarrow F - f_k$
end-if
end-begin
until Pr' is complete
end. {COM_MIN}

Algorithm 5.2 *PHORIZONTAL*
input: R: relation; Pr: set of simple predicates
output: M: set of minterm fragments
begin
$Pr' \leftarrow$COM_MIN(R, Pr)
determine the set M of minterm predicates
determine the set I of implications among $p_i \in Pr'$
for each $m_i \in M$ **do**
if m_i is contradictory according to I **then**
$M \leftarrow M - m_i$
end-if
end-for
end. {PHORIZONTAL}

Example 5.11

Derived Horizontal Fragmentation

Derived fragmentation is used to facilitate the join between fragments.

In some cases, the horizontal fragmentation of a relation cannot be based on a property of its own attributes, but is derived from the horizontal fragmentation of another relation.

(M)

Benefits of Derived Fragmentation

PAY (TITLE, SAL)

EMP (ENO, ENAME, TITLE)

Primary Fragmentation:

$$PAY_1 = \sigma_{SAL \leq 30,000}(PAY)$$

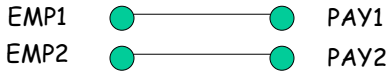
$$PAY_2 = \sigma_{(SAL > 30,000)}(PAY)$$

Using Derived Fragmentation:

EMP_1 = EMP SJ PAY_1

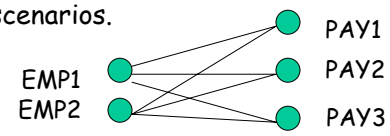
EMP_2 = EMP SJ PAY_2

Simple Join Graph between EMP and PAY



EMPi and PAYi can be allocated to the same site.

Not using derived fragmentation: one can divide EMP into EMP1 and EMP2 based on TITLE and divide PAY into PAY1, PAY2, PAY3 based on SAL. To join EMP and PAY, we have the following scenarios.



More communication overhead !

(M)

Derived Fragmentation

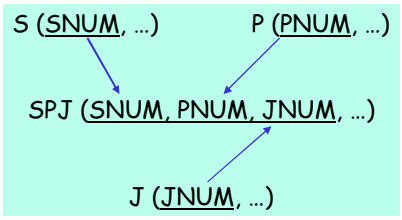
EMP (ENO, ENAME, TITLE) PROJ (PNO, PNAME, BUDGET)

EMP_PROJ (ENO, PNO, RESP, DUR)

- How do we fragment EMP_PROJ ?
  - Semi-Join with EMP, or
  - Semi-Join with PROJ
- Criterion: Support the more frequent join operation.

(M)

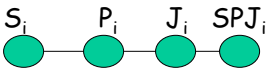
Star Relationships



- Design the primary horizontal fragmentation for SPJ.
- Derive the derived fragmentation designs for S, P, and J accordingly.
  - $S_i = S \ SJ_{SNUM} \ SPJi$
  - $P_i = P \ SJ_{PNUM} \ SPJi$
  - $J_i = J \ SJ_{SNUM} \ SPJi$

Exception Case: Primary fragmentation on member relation.

How does the join graph looks like when joining all the relations?



(M)

Chain Relationships

- Design the primary fragmentation for R1 into M fragments.
- Derive the derived fragmentation for Rk as follows:
  - $Rk_i = Rk \ SJ_{RkFK=R(k-1)PK} \ R(k-1)i$ for $2 \le k \le N$ in that order for all $1 \le i \le M$.