Lecture 3

Distributed Database Design

Outline

- Introduction
- Background
- Distributed Database Design
    - Fragmentation
    - Data distribution
- Database Integration
- Semantic Data Control
- Distributed Query Processing
- Multidatabase Query Processing
- Distributed Transaction Management
- Data Replication
- Parallel Database Systems
- Distributed Object DBMS
- Peer-to-Peer Data Management
- Web Data Management
- Current Issues

Design Problem

- In the general setting :

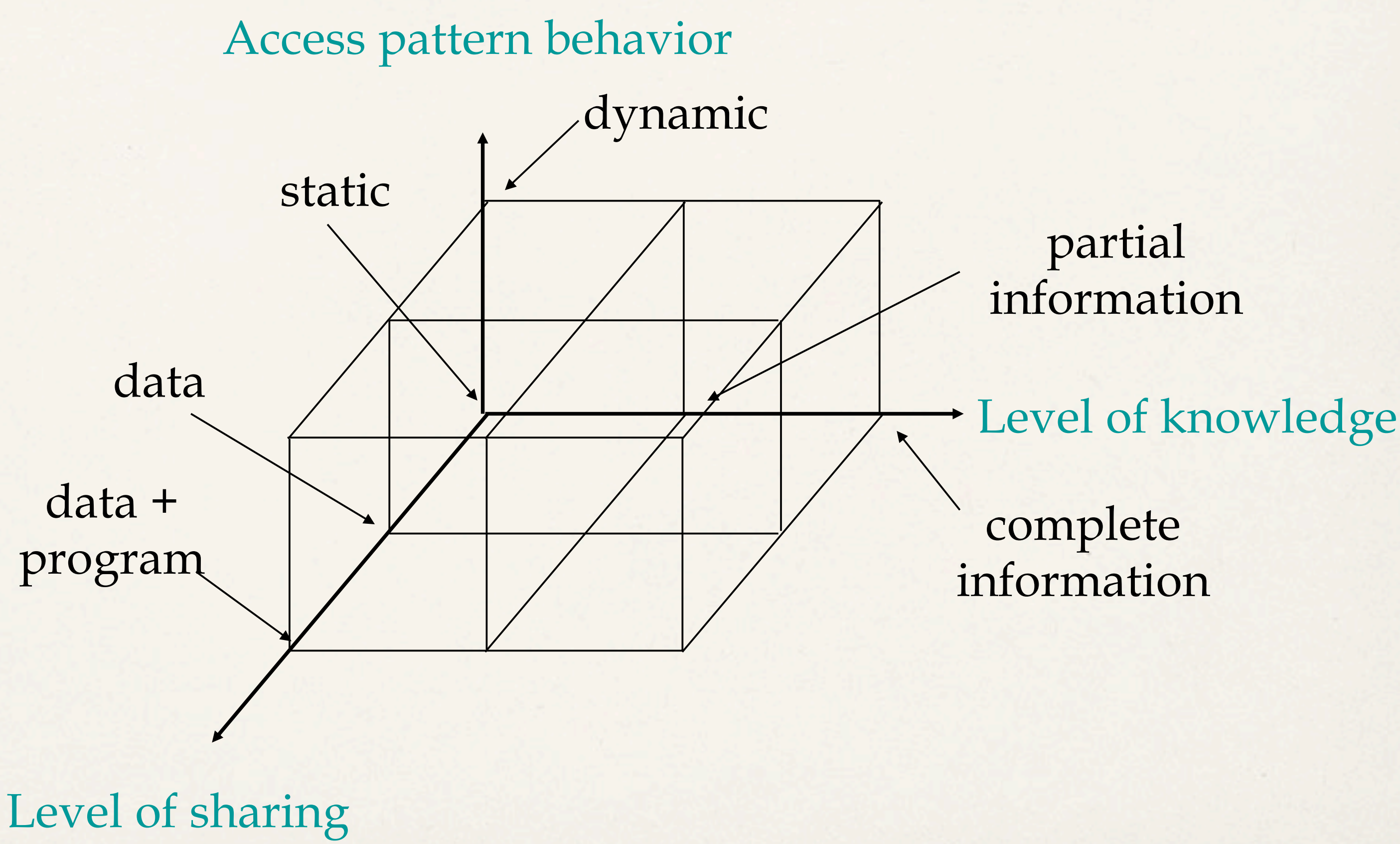
  Making decisions about the placement of data and programs across the sites of a computer network as well as possibly designing the network itself.

- In Distributed DBMS, the placement of applications entails

  placement of the distributed DBMS software; and

  placement of the applications that run on the database

Dimensions of the Problem



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

Top-Down Design

1. Requirements Analysis:
defines the environment of the system and "elicits" both the data and processing needs of all database users

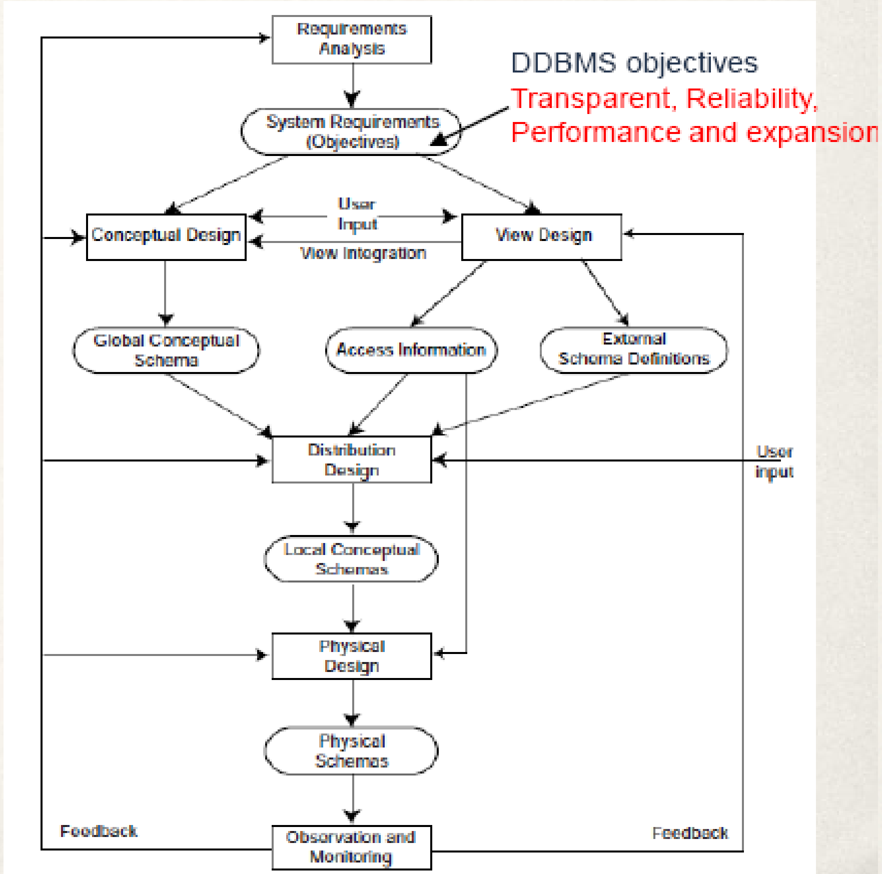
2.Conceptual Design:
determine entity types and relationships among these entities

3. View Design:
Define the interfaces for end users.

4. Distribution Design:
Its objective is to design the local conceptual schemas (LCSs) by distributing the entities over the sites of the distributed system, it includes Fragmentation and data allocation processes.

5. physical design:
maps the local conceptual schemas to the physical storage devices available at the corresponding sites.

6.Observation and Monitoring
Include constant monitoring and periodic adjustment and tuning



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 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

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

Degree of Fragmentation

finite number of alternatives

tuples
or
attributes

relations

Finding the suitable level of partitioning within this range

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 $\dfrac{\textit{read - only queries}}{\textit{update quries}} \gg 1$ 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
REALITY	Possible application	Realistic	Possible application

Information Requirements

- Four categories:

  Database information

  Application information

  Communication network information

  Computer system information

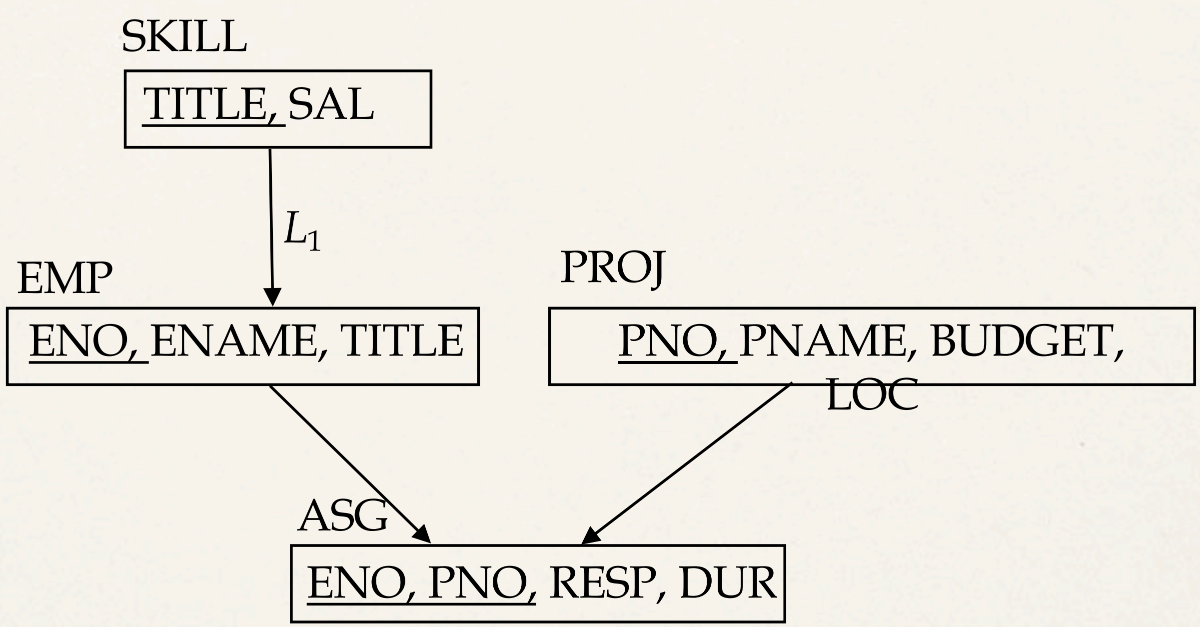
Fragmentation

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

PHF – Information Requirements

- Database Information

    relationship



    cardinality of each relation: $card(R)$

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" $\wedge$ BUDGET≤200000

m_2: **NOT**(PNAME="Maintenance") $\wedge$ BUDGET≤200000

m_3: PNAME= "Maintenance" $\wedge$ **NOT**(BUDGET≤200000)

m_4: **NOT**(PNAME="Maintenance") $\wedge$ **NOT**(BUDGET≤200000)

PHF – Information Requirements

- Application Information

  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: $acc(q_i)$

  - The frequency with which a user application *qi* accesses data.
  - Access frequency for a minterm predicate can also be defined.

Primary Horizontal Fragmentation

Definition :

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

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

Therefore,

A horizontal fragment R_i of relation R consists of all the tuples of R which satisfy a minterm predicate m_i.

$$\Downarrow$$

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.

Preliminaries :

Pr should be *complete*

Pr should be *minimal*

Completeness of Simple Predicates

- A set of simple predicates *Pr* is said to be *complete* if and only if 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.

- Example :

  Assume PROJ[PNO,PNAME,BUDGET,LOC] has two applications defined on it.

  Find the budgets of projects at each location. (1)

  Find projects with budgets less than $200000. (2)

Completeness of Simple Predicates

According to (1),

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

which is not complete with respect to (2).

Modify

Pr ={LOC="Montreal",LOC="New York",LOC="Paris",
BUDGET≤200000,BUDGET>200000}

which is complete.

Minimality of Simple Predicates

- If a predicate influences how fragmentation is performed, (i.e., causes a fragment f to be further fragmented into, say, f_i and f_j) then there should be at least one application that accesses f_i and f_j differently.
- In other words, the simple predicate should be *relevant* in determining a fragmentation.
- If all the predicates of a set Pr are relevant, then Pr is *minimal*.

Minimality of Simple Predicates

Example :

Pr ={LOC="Montreal",LOC="New York", LOC="Paris",

BUDGET≤200000,BUDGET>200000}

is minimal (in addition to being complete). However, if we add

PNAME = "Instrumentation"

then *Pr* is not minimal.

Allocation Model

- Constraints

  Response Time

  execution time of query $\leq$ 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