ADVANCED TOPICS IN DATABASES

Distributed Databases

Distributed Database Design

- **Top-down**
  - mostly in designing systems from scratch
  - mostly in **homogeneous systems**
  - division of the DB by the different nodes
- **Bottom-up**
  - when the databases already exist at a number of sites
  - integration of multiple DBs into a global DB distributed across multiple machines
    - **Heterogeneous environment plus difficulties**

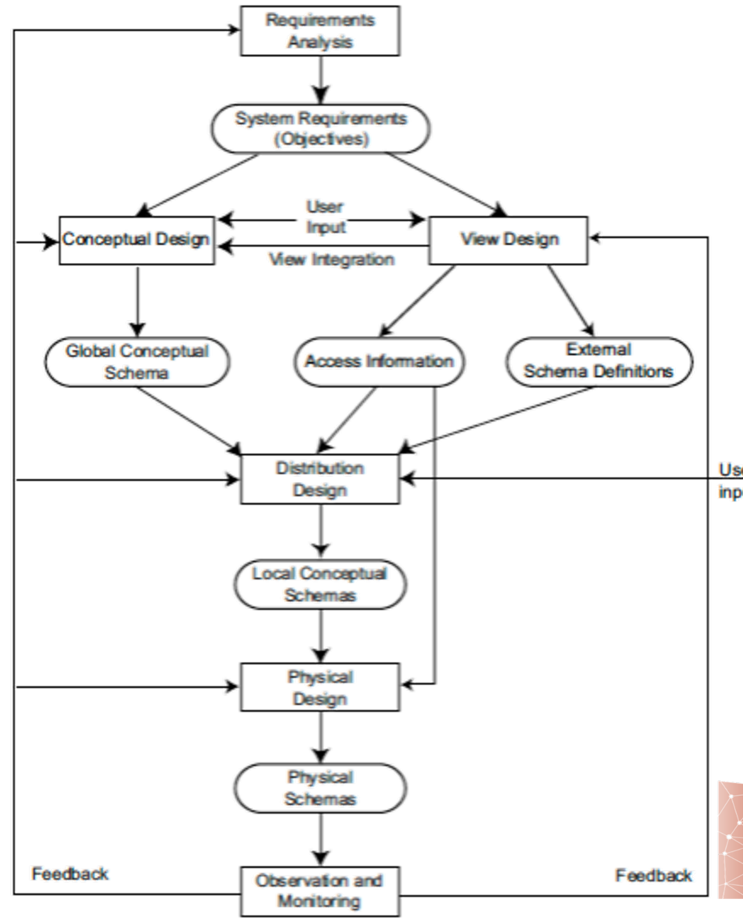
Top-Down Design Process

The **conceptual design** , on the other hand, is the process by which the enterprise is examined to determine entity types and relationships among these entities.

Entity analysis is concerned with determining the entities, their attributes, and the relationships among them.

Functional analysis is concerned with determining the fundamental functions with which the modeled enterprise is involved.

Global conceptual schema (GCS): to design the local conceptual schemas (LCSs) by distributing the entities over the sites of the distributed system.

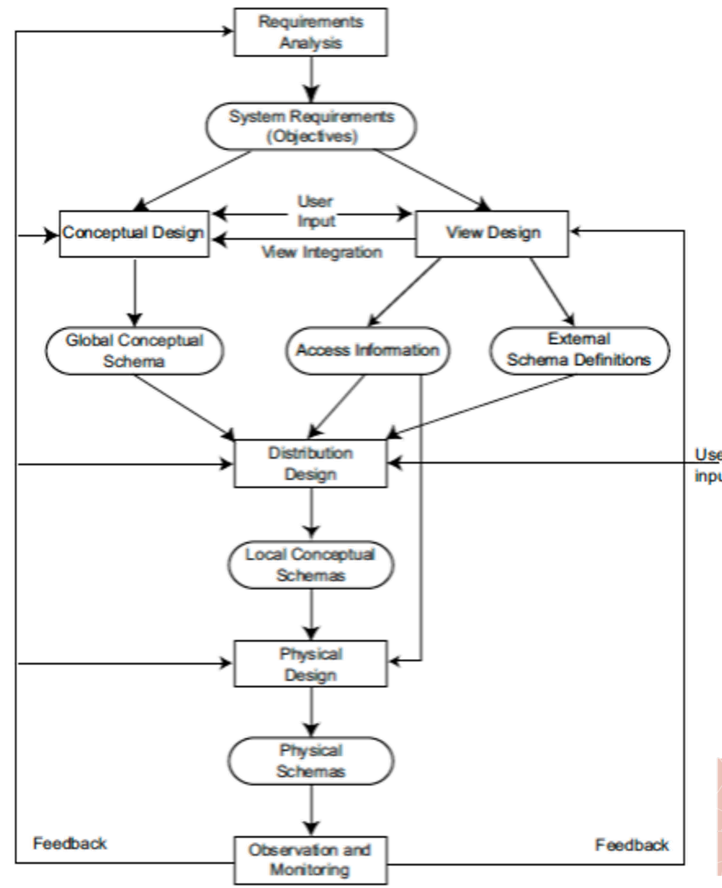


Top-Down Design Process

The distribution design: It consists of two steps: fragmentation and allocation.

Physical design: The last step in the design process, maps the local conceptual schemas to the physical storage devices available at the corresponding sites. The inputs to this process are the local conceptual schema and the access pattern information.

Last Step: The **result is some form of feedback, which may result in backing up to one of the earlier steps in the design.**



Top-Down Design Issues

How to distribute the tables that support the global conceptual data model across the various nodes of the network?

There should be cooperation between the different nodes.

The General Idea→ Maximize local processing

"Data should be put in front of those who need it most"

--» Which local tables?

– i.e. which will reside entirely in a single node and to which node they correspond?

--» Which tables to shard? And where are they stored?

--» Which tables or shards we must duplicate? and on which nodes should we duplicate?

Top-Down Design

The distribution of data across the various nodes **aims to maximize local processing.**

Analyzing:

- the most frequent transactions, what data they access and on which nodes they originate

Find the best configuration for data location

- Another approach is to continuous redistribution of data across the various nodes.

  - the system evolves over time => leads to a redistribution of data. The initial data may no longer be the most appropriate.
  - Location should not be static.
  - Based on **a transaction history. The shards shou**ld be **dynamically transferred between the nodes by the system** in order **to maintain the most appropriat**e distribution.

Exercise

Database

EMP ={**codemp**, nome, salario, ... coddep}

DEPT={ **coddep,** nomedep, localização}

PROJ={ **codproj,** fduracao,coddep }

PROJ_EMP ={ **codproj, codeemp**, }

The following are requested:
1 - DB fragmentation scheme;
2 - DB allocation scheme.

A company has 3 DEPTS (1, 4, 5) that are geographically distributed and reside across the company's 3 network sites. Suppose that site1 is central and contains the entire database, but sites 2 and 3 only access data from DEPTS 4 and 5 respectively and even if:

- DEPT 4 only wants to access name, employee code and participation in projects;
- DEPT 5 accesses all data on employees and participations.

1- Obtain the name of employees and their dependents for employees working in DEPT5;

2 – Obtain the names of employees who work in DEPT5 and who participate in projects that are not coordinated by DEPT5.

PROCESSING AND OPTIMIZATION

Two aspects what distinguishes the process of optimizing issues in distributed systems from centralized systems:

- the use of means of communication

- taking advantage of the parallelism inherent in distributed systems.

Attention to transactions involving data located in different nodes.

The optimization problem will have to:

minimize, in terms of resources, the total cost of processing the issue by minimizing the amount of traffic on communication networks;

Minimize response time, taking advantage, whenever possible, of the processing being carried out locally to the node where the data is stored.

PROCESSING AND OPTIMIZATION

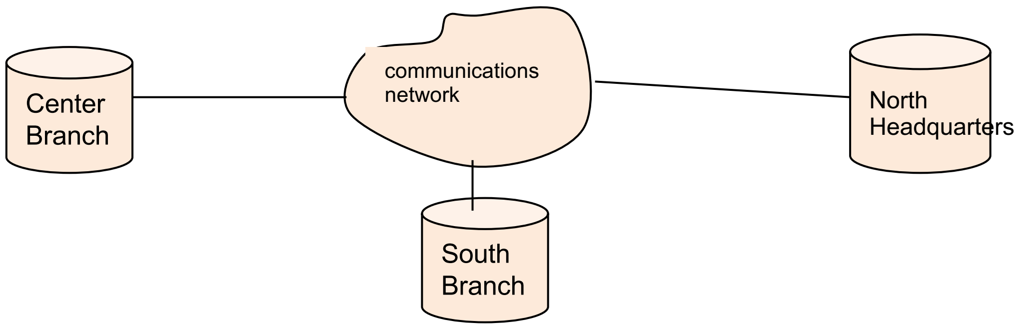
If the issue cannot be resolved locally, it is necessary to divide its processing among the nodes involved.

The implementation of a Global issue consists (normally) in a set of local sub-sets runs, probably in parallel, whose intermediate results are moved between participating nodes, and, at last gathered to constitute the final answer.

PROCESSING AND OPTIMIZATION

EXAMPLE

Let's consider a company with headquarters in the North and branches in the Center and South



At any given point in time, the distribution of data and processing is as follows:

Product –> Some attributes(cod_prd, designation, price) are duplicated in each node

Stock = Stock_North + Stock_Center+ Stock_South -> distributed across the three nodes

Customer –> distributed across the three nodes

Cust = Cust_North + Cust_Center + Cust_South

PROCESSING AND OPTIMIZATION

Supplier –> located at the North node

Order >– Locally managed

Ord = Ord_North + Ord_Center + Ord_South

Invoice –> located in north node

- **Question** asked at the Centre's branch
  "Who are the customers (num_cli, name) of the Southern branch with overdue invoices. What are those invoices (num_fact)?"

Suppose the CUSTOMER AND INVOICE tables are:
CUSTOMER (num_cust, name, address, zone, …).
INVOICE (num_inv, num_cust, date, state_invoice, …)

(North, Center, South)

(Paid, not_Paid)

and that the optimization criteria only involves minimising of communications costs

PROCESSING AND OPTIMIZATION

- The question could be put as follows

SELECT C.num_cust, C.name, I.num_inv

FROM Customer C, invoice I

WHERE C.zone = ‘South’ AND F.state_invoice = ‘Not_Paid’ AND C.num_cust = I.num_cust

In relational algebra it will be:

$$\text{TEMP1} = (\text{INVOICE} \bowtie \text{CUSTOMER})$$

$$\text{TEMP2} = \sigma_{\text{state\_invoice} = \text{“Not\_Paid”} \wedge \text{zone} = \text{“South”}}(\text{TEMP1})$$

$$\Pi_{\text{num\_cust, name, num\_inv}}(\text{TEMP2})$$

which may give rise to the following sub-questions:

PROCESSING AND OPTIMIZATION

Optimized

Q1 TEMP1 = Π num_cust, name ($\sigma_{zone= \text{"South"}}$ (CUSTOMER))

Q2 TEMP2 = Π num_inv, num_cust ($\sigma_{STATE\_invoice\ =\ \text{"Not\_Paid"}}$(INVOICE))

Q3 Π num_cust, name, num_inv (TEMP1 $\bowtie$ TEMP2)

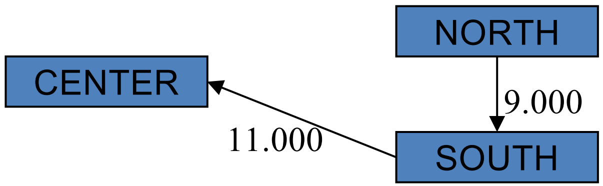
And further assume:

- the total number of South customers is 1000.
- pair (num_cust, name) **spend 50 bytes**, immediately TEMP1 spend **50.000bytes** (1000 x 50)
- there are 600 invoices unpaid;
- pair (num_cust, num_inv) **spend 15 bytes**, immediately TEMP2 spend **9.000 bytes (600 x 15)**.
- in the South zone there are 200 unpaid bills.
- pair (num_cust, name, num_inv) **spend 55 bytes**, immediately there will be a total of **11.000 bytes (200 x 55).**

"What's the best strategy to solve the problem?"

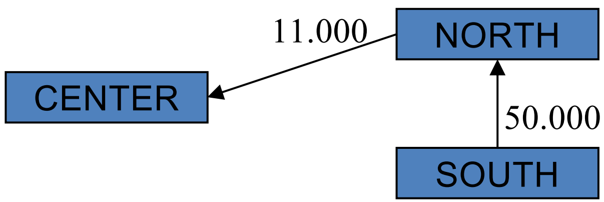
PROCESSING AND OPTIMIZATION

FIRST STRATEGY



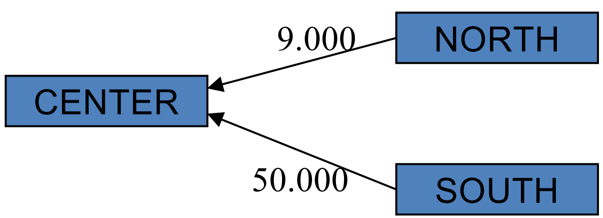
→ Q1 in the South
→ Q2 in the North and pass TEMP2 to the South
→ Q3 in the South and send the result to Center
20.000 bytes

SECOND STRATEGY



→ Q2 in the North
→Q1 in the South and pass TEMP1 to the North
→ Q3 in the North and send the result to Center
61.000 bytes

THIRD STRATEGY



→ Q1 in the South and pass TEMP1 to Center
→ Q2 in North and pass TEMP2 to Center
→ Q3 in the Center
59.000 bytes

PROCESSING AND OPTIMIZATION

In terms of communications, the most cost-effective strategy is

FIRST STRATEGY

CONCLUSION:

- On a **1st lev**el, there is the optimization of the **global issue**, with the division into sub-questions, Selection of nodes responsible for the execution of these sub-questions and **the selection of the best strategy for this execution.**

- On a **2nd level t**here is a **optimization of each sub-issue in the node where it is executed.**