**Case Study: Airline Reservation System**

An Airline Reservation System stores the following information   
(The following description may lead to un-normalized relations normalize them whenever required).

The Flight Details   
It includes the originating flight terminal and destination terminal, along with stops in between, number of seats booked/available seats between two destination etc.   
  
Customer Description   
It includes customer code, name, address and phone number. This information may be used for keeping the records of customer for any emergency or for any other kind of information   
  
Reservation Description   
It includes customer code number, flight number, date of booking, date of traveling, (You may assume any other field/relation, if needed).

Let us assume that this distributed system is to be used following application:

a. A request for booking/cancellation of flight from any source to any destination, giving connected flights in case no direct flight between the specified Source-Destination pair exists.

b. Calculation of high fliers (most frequent fliers) and calculating appropriate reward points for these fliers.

Our design includes:

i) The global schema, fragmentation schema, allocation schema.

ii) SQL commands for above queries/applications

iii) How the response for application 1 and 2 will be generated.

iv) Implementation of the database using a centralized database management system

DISTRIBUTED DATABASE:

Distributed database implies that a single application should be able to operate transparently on data that is spread across a variety of different databases and connected by a communication network.

A distributed database system consists of a collection of sites, connected together via some kind of communications network, in which

1. Each site is a database system site in its own right,

2. The sites have agreed to work together so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the users own site.

It follows that the so-called distributed database is really a kind of virtual objects, whose component parts are physically stored in a number of distinct real database at a number distinct sites.

A distributed database system consists of a collection of sites, each of which may participate in the execution of transactions which access data at one site, or several sites. The main difference between centralized and distributed database systems is that, in the former, the data resided in one single location, while in the latter; the data resided in several locations.

Each site is able to process local transactions, those transactions that access data only that single site. A site may participate in the executions of global transactions, those transactions that access data in several sites. The execution of global transactions requires communication among the sites.

Each site is a database system site in its own right. In other words, each site has its own local real databases, its own local users, its own local DBMS and transaction management software and its own local data communications manager. The distributed database system can thus be regarded as a kind of partnership among individual local DBMS at the individual sites.

In the given context there are four sites, i.e., Delhi, Mumbai, Chennai and Kolkata and these sites are geographically dispersed. Each site has its own computer with a database consisting of all the customers and flight details maintained at that site. The database systems at the various sites are based on the relational model. Each branch maintains the relation reservation where Reservation\_scheme=(site\_code,fl\_no,cust\_code,travel\_date)

Site containing information about the four sites maintains the relation (Cite\_scheme), where

Cite\_scheme=(Cite\_code,Cite\_name)

A local transaction is a transaction that accesses information in the one single site, at which the transaction was initiated. A global transaction, on the other hand is one which either access accounts in a site different from the one at which the transaction was initiated, or access several different sites.

Then the advantages are surely obvious: The distributed arrangement combines efficiency of processing (the data is stored close to the point where it is most frequently used) with increased accessibility (it is possible to access a Delhi information from Kolkata and vice versa, via the communication network), reliability and availability and speedup of query processing.

Allowing the structure of the database to mirror the structure of the enterprise is probably the number one advantage of distributed systems. There are some disadvantages too, of which the biggest is the fact that distributed systems are complex, software development cost, greater potential for bugs and increase processing overhead.

As per commercial implementations, most of todays relational products offer some kind of distributed database support. Some of the best known include,

(a) INRESS/STAR , from The ASK Group Inc.'s Ingress Division;

(b) the distributed database option of SQL Server

It is worth pointing out that all of the systems listed above, both prototypes and products, are relational.

**OBJECTIVES:**

1. Local autonomy

2. No reliance on a central site

3. Continuous operation

4. Location independence

5. Fragmentation independence

6. Replication independence

7. Distributed query processing

8. Distributed transactions management

9. Hardware independence

10. Operating system independence

11. Network independence

12. DBMS independence

These above twelve objectives are not all independent of one another, nor are they necessarily exhaustive, nor are they all equally significant. However, they are useful as a basis for understanding distributed technology and as a framework for characterizing the functionality of specific distributed systems.

