

M. Tech. (Computer Science) dissertation Series

Domestic Airlines Tour Planning Software

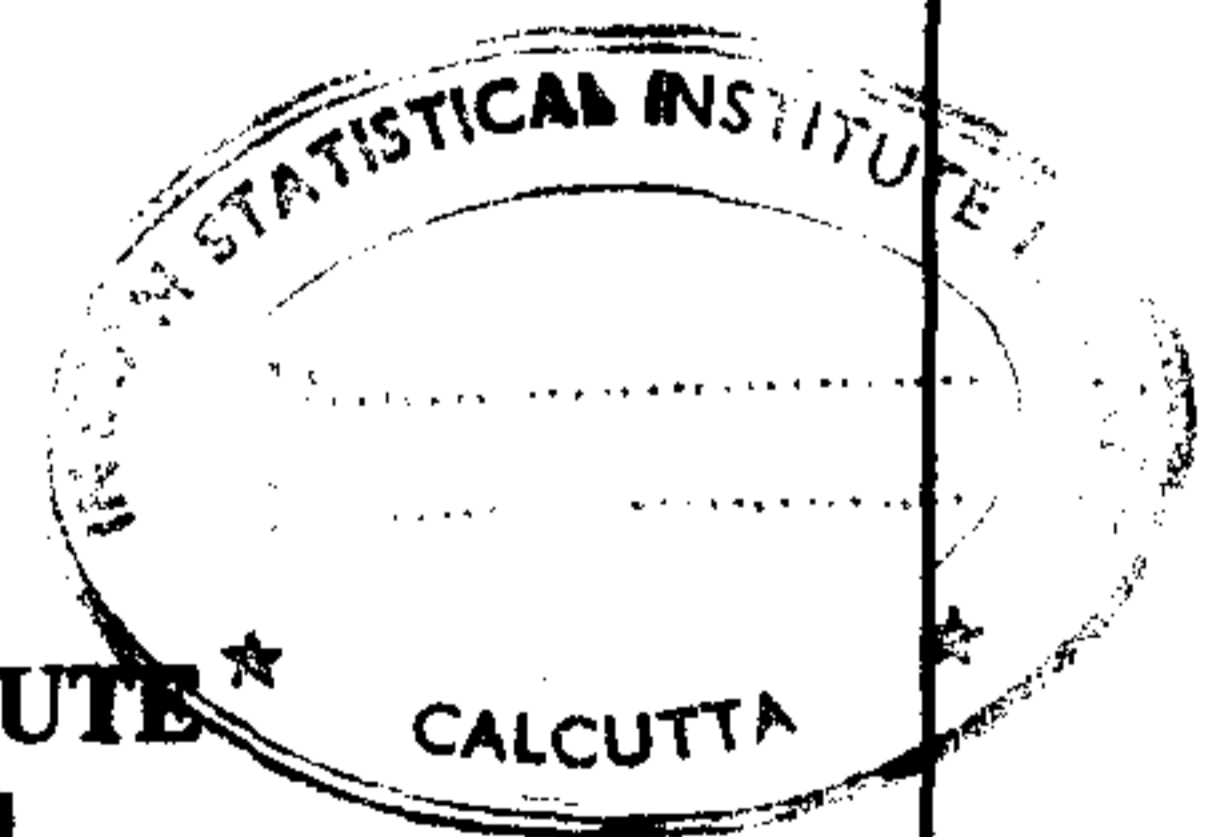
**a dissertation submitted in partial fulfilment of the requirements
for the M. Tech. (Computer Science) degree of the
Indian Statistical Institute**

By
Harekrushna Patel

under the supervision of
Probal Sengupta
(Computer Vision and Pattern Recognition Unit)

INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road,
Calcutta - 700 035

31st July 1998




Certificate of Approval

This is to certify that the dissertation entitled **Domestic Airlines Tour Planning Software** submitted by **Mr. Harekrushna Patel**, towards partial fulfilment of the requirement for the Master degree of Technology in Computer Science at the Indian Statistical Institute, Calcutta, embodies the work done under my supervision. His performance in the said project is satisfactory and has achieved good results. I wish him all success in his future endeavours.

Date : 31st July 1998

Susmita Das-Kalra
(External examiner) 31.8 '98

 31/8/98
(Probal Sengupta)
Professor
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
Calcutta

ACKNOWLEDGEMENTS

I am expressing my sincere regards, to my project co-ordinator and guide, Prof. Dr. Probal Sengupta, whose deep involvement in the project, inspiration, strict schedule, love and affection towards me, enabled me to complete this project in due course of time. During the development and implementation stages, his timely issuance of approaches, ideas and lessons have helped me to simplify the development to a great extent.

I thank each individual of the CVPR Unit for providing me all kind of help in many different ways.

I would like to take this opportunity to thank my teachers for their excellent teaching during the M. Tech. course, which helped me in this work.

It is very much difficult to acknowledge all individuals, who are directly or indirectly involved, but I especially thank Mr. Rajeev Verma, Mr. Rajesh Babu, Mr. Biplab Sarkar and Mr. Parveen Gupta for their helpful suggestions on the occasions and providing me resources to complete this work.

At this moment, I would like to acknowledge my friends Mr. Balakrishna Reddy, Mr. Sudhakar Rao, Mr. Rajeev Sinha, Mr. Pawan Kumar Singal, Mr. Manas Ranjan Jagdev, Mr. Piyush Ranjan Kumar and all my classmates, who shared those moments of joy and frustration and made my two years stay at ISI enjoyable.

I shall pay my gratitudes towards my parents, brother-in-law Mr. Baldevbhai Patel, cousin Mr. Govindbhai Patel and my elders whose unseen blessings, encouragements and support can't be measured in words.

Harekrushna Patel
31st July 1998

Contents

Chapter	Topic	Page no.
	Certificate of Approval	1
	Acknowledgement	2
	Abstract	4
1	Introduction and Problem Specifications	5
	1.1 Motivation	5
	1.2 Problem Specification	5
2	Problem Analysis and A Proposed Solution	7
3	Design and Implementation of The System	10
	3.1 Design Issues	10
	3.2 Data Structures	13
	3.3 Algorithms	16
4	The User Guide	20
	4.1 The Installation	20
	4.2 The Maintenance	20
5	Results and Conclusion	21
	5.1 Results	21
	5.2 Conclusion	22
	References	23

ABSTRACT

In this project, a system is considered that can plan a tour within India through domestic airlines. The administrator can maintain the system. Normal users can plan a tour using the system.

To make the system versatile, the application is made text based. The original data about domestic airlines is available on the Internet. The database required for the system can be created from the data and can be updated by the administrator at certain time intervals. This database can reside on the system itself.

The application is planned to be hosted on an Internet World-wide-web site so that anyone with an access to the internet can use the software to plan his or her tour.

1. Introduction and Problem Specifications

1.1 Motivation

During the past few years, India has grown as a global economy in the world. It has achieved growth in the industry. People therefore need to travel from one place to another as quickly as possible, for purposes related to industry, business or any other. Air travel is undoubtedly the fastest mode of travel. With the increase in industrial and business activities, the number of persons travelling by air in India has increased significantly. Air travel in India is still a costly alternative, given the high fares of travel and the relatively low income of average Indians. However, under certain circumstances, a traveller may not mind the extra fare provided certain other requirements are fulfilled. Considering an average Indian air traveller, his/her requirements could be:

1. Having to pay the minimum amount in air fare.
2. Completing the journey in the minimum possible time.
3. Having to wait at airports of intermediate stoppages for a minimum amount of time.

In the earlier days of less (air) travel an average traveller used to fly only from one of the few major cities to another. But nowadays, the incidents of travellers having to fly from a smaller city to a major city, or vice versa or between two smaller cities, is on the rise. However, the airlines infrastructure in the country is not sufficiently geared up to provide direct flights even for a small fraction of such 'non-standard' journeys. A traveller making such a non-standard journey therefore has to rely upon his/her own judgement (or, in actual practice, on the judgement of the travel agent) regarding the flight pattern given the flight schedules of the different airlines companies plying in India.

The present project is aimed at creation of a software that can act as an aid to a person wanting to **plan a travel or tour in India, through domestic airlines.**

1.2 Problem Specification

The title of this project is "**Domestic Airlines Tour Planning Software**". The problem is to design a system, which have database about all domestic airlines within India. The system should be able to reply the users' queries.

The queries are related to plan a tour within India through domestic airlines. The possible queries are:

1. Query about a flight.
2. Query about a city having an airport.
3. Query about list of flights available to travel from one place to another with certain constraints in mind (like minimum cost, minimum time, minimum waiting time during the travel, fixed time interval to start the travel etc. or any possible combination of these constraints).
4. Query about the estimate of fare, time required to travel from one place to another.

The original database about domestic airlines is available on the Internet. The database required for the system can be created and reside on the system. This database can be updated at certain time of interval and is used to reply the users' queries.

Some of the airlines companies (for example, Jet Airways, <http://www.jetairways.com>) even maintain a query system similar to the ones talked about here but for their own flights only. However, to the best of our knowledge, no airlines company offers a service of the sort talked about here where:

- All airlines companies are taken into consideration.
- Relative weightages among at least three constraints can be stipulated by the user.

2. Problem Analysis and A Proposed Solution

From the specification of the software, it is clear that:

- The 'raw' data for the software is acquired from the database of flight schedules maintained by different airlines companies. The databases are assumed to be 'on-line' and accessible at all times through the Internet.
- Different airlines companies may maintain their databases in their own chosen schema format. However, insofar as our software is concerned, the queries that are allowed must be in a uniform format.
- The most important type of query answered by our system is that of constraint based route planning spanning multiple airlines. Such a query may require some non-trivial computations to be carried out to arrive at the result.
- It is worthwhile carrying out certain types of computations only at longer intervals and store the results of such computations locally.
- While data of flight schedules are not likely to change very frequently, they would change sometimes. Under such circumstances, the above mentioned computations must be repeated (at least partially) to keep local data consistent.
- A special user should be there who decides when to trigger the recomputation of local data.
- Normal users should remain oblivious of above mentioned administration duties.

Some definitions of important components of our software may be in order.

The global database

The global database is our abstraction of a host of flight databases maintained by individual airlines companies operating in India. Of course, each airlines have its own style and organization of the database. However, considering all airlines companies together, the following common set of information is always available in the public domain:

A Flight Number that is associated with the day (of week), time of departure from the starting airport, intermediate hops (giving the intermediate stoppage airport, day and time of arrival and day and time of departure) and finally, the day and time of arrival at the final destination. We consider the above above mentioned abstraction of flight data of all companies as the global database.

The local database

To reply users' queries, it may be necessary to compute certain things and, given the nature of the queries, some of the computations are likely to be repeated again and again. For example, suppose the user wants to know the shortest path to travel from one place to another with certain constraints in mind, the system would have start from all practically feasible paths between the places. In a later query involving the same two places but with possibly different constraints, the same practically feasible paths need to be computed again. However, the actual number of airports in India is not very large and a one-shot computation (or "pre-computation") of all practical feasible paths between all possible pairs of start and destination points, is quite feasible computationally. Of course, the result of such a computation may get invalidated by the introduction/withdrawal of particular flights or even by rescheduling of flights. Since such changes are not very frequent, it may be worthwhile to carry out the computations on the available data and store the same in a local database. The Administrator can maintain this database by periodically refreshing its contents through repeat computation on the most current airline data in the global database.

User queries will always be replied using the local database.

The database handler

The global database will of course change with time. Such changes necessarily update the local database to keep the local database consistent with the global database at all times. As per our assumptions, the global database consists of several actual databases available on the Internet and these databases may have different formats. In our system, we propose an abstraction called the database handler that is responsible for providing a uniform monolithic view of the data (pertinent to the queries mentioned earlier) of all airlines companies taken together.

One major task of the database handler is to upgrade the local database to keep the computed paths stored in it consistent with the global database. In our proposed system, we assume the existence of a special user called the 'Administrator' who triggers the upgradation. For the administrator, our software provides with the necessary interfaces to trigger the upgradation. The special options for the same are protected from normal users through a password mechanism.

The query processor

On the user side, the user can use the system to plan the travelling in India through domestic airlines. To plan a travelling, he can have queries. The system can reply these queries and for that, both the local and global databases need to be accessed. The component that deals with the queries, is called a query processor. The

user's query may make the query processor access either or both the local and the global databases.

Thus, the overall situation is shown in the figure below.

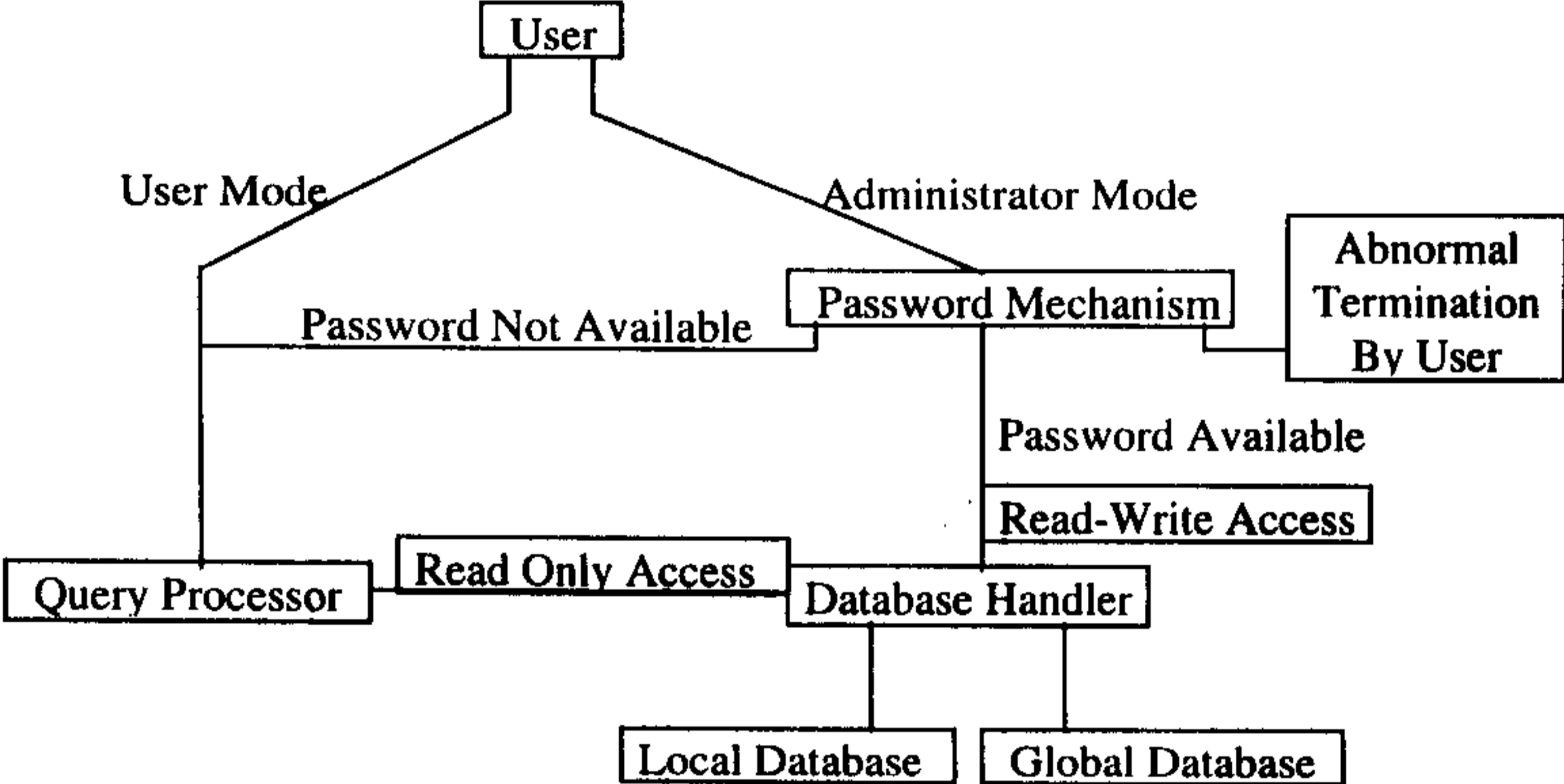


Fig. The overall view of the proposed system.

3. Design And Implementation Of The System

3.1 Design Issues

The possible queries are:

1. To know about a particular flight.
2. To know about a particular city having an airport.
3. To know about a particular airport.
4. To know the shortest path from one place to another with certain constraints like minimum time, minimum waiting time, minimum fare, fixed time and date to start the journey or any possible combination of these constraints.
5. To estimate the time, waiting time, fare for journey from one place to another.

We assume that the global database consists of several databases available at various sites on the Internet, one for each airlines company (that is, each airlines company has a site having database about it).

For a query of type 1, 2 or 3, no computation is needed. For a query of type 4 and 5, we need to compute all *practically feasible paths* (see below) from one place to another or to estimate the time, waiting time, fare. To avoid repetition of feasible path computation, we use a local database consisting of all practically feasible paths and the estimation of time, waiting time and fare required for journey for all possible pair of source and destination. Further, the global database stores data for all airlines companies. The local database has access to data about all cities, airports, and general data about airlines through the global database.

Practically feasible path

To travel from one place to another, it may be possible that direct flight is not available. The traveller may have to change flights, i.e., the journey may have hops. We feel that a path having a large number of hops is impractical, even if it takes less time than other paths. We have made an assumption that one can travel comfortably from one place to another making at most three hops.

We have also made a few other assumptions taking into consideration the ease and comfort of travel. The assumptions are:

- Any practically feasible path has at most three hops. That is, it consists of at most four flights.
- A traveller will not wait for any (intermediate) flight for more than one day.
- A flight may have stoppages in between. However, the number of stoppages between source and destination will not be more than four.

The query processor

The query processor can process the following possible queries:

1. Query about a particular flight. Serviced by querying the appropriate Web-based database of the concerned airlines company. Airlines companies' data are abstracted by the global database.
2. Query about a particular city having an airport. This pertains to basic information about the city and/or its airport(s). We assume that data for answering queries would also be available online and therefore be abstracted by the global database.
3. Query to get the list of flights available to travel from one place to another with certain constraints like:
 - want to travel in minimum time
 - want to travel with minimum waiting time
 - want to travel with minimum fare
 - want to start the journey in fixed time interval
 - any possible combination of the constraints given above

This query is the major concern of our design and would be answered using the local database containing all possible pre-computed practically feasible paths.

4. Query to estimate the time, waiting time and fare required to travel from one place to another. This query is actually an extension of the previously mentioned query whose results come as a by-product of the computations involved therein.

In the work reported in this thesis, we could not create a properly functioning global database. The main reason was that till very late stage of our work, a reasonably fast access to the internet was not available. Also, many of the airlines company operating in India has still not made their flight information available on the Internet that can be effectively accessed. Nevertheless, we proceed with an abstract design of the (global) database with the expectation that the present work can be

practically extended to properly connect to components of the global database as and when data of airlines are accessible through the Internet.

The figure given on the next page describes both the databases altogether by an **E-R relationship diagram**.

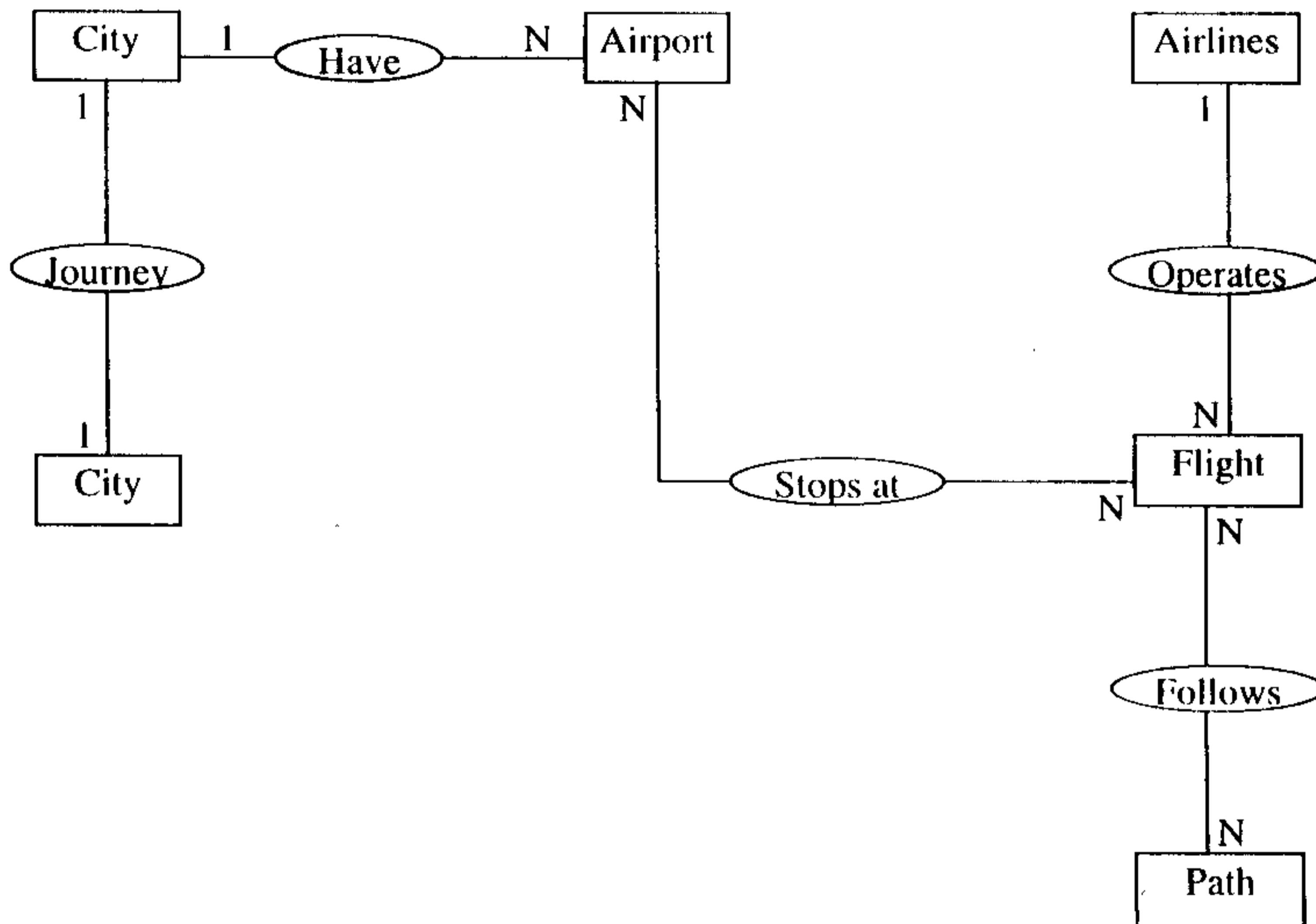


Fig. The E-R relationship diagram.

Entity attributes

1. City : City id, City name, State, Status.
2. Airport : Airport id, Airport name, Airport address, Airport city id, Level of operation.
3. Airlines : Airlines id, Airlines name, Airlines I-net address.
4. Flight : flight id, Source airport id, Destination airport id, Days of operation flag, Departure time, Arrival time, Number of stopages, Array of airports (stopages) including source and destination, Associated airlines id, Fare, Capacity.
5. Path : Source city id, Destination city id, Departure time, Days of operation path flag, Number of flights in path, Arrays of flight information (flight information

includes flight id, departure time, arrival time, where to catch a flight, where to leave it), total time, waiting time, total fare.

Relation attributes

1. Stops at : Associated airport id, Arrival time, Departure time, Previous stoppage airport id, next stoppage airport id, Time stays at stoppage.
2. Journey : Source city id, Destination city id, Estimate of time required, Estimate of waiting time, Estimate of fare.

3.2 Data Structures

An “in-memory” view of the various entities can be described by the following data structure.

1. Flight

Deals with the data of a flight. The contents are:

- Flight ID
- Source Airport ID
- Destination Airport ID
- Departure Time
- Arrival Time
- Days of Operation FLAG
- Number of Stoppages
- Array of Stoppages
- Route Number
- Type of the Aircraft
- Airlines ID
- 2-dimensional Array of Fare-structure. There is one entry for each pair of stoppages. The $i \times j$ -th entry contains information of fare from stoppage i to stoppage j . A Fare-structure element stores a) fare for *the economy class*, b) fare for *the executive class* and c) fare for *the student concession class*.
- Capacity of Flight

Note : If a new class of journey is introduced at a later data, we can add a fare for the new class to the Fare-structure.

2. Airport

The contents are :

- Airport ID
- Name
- Address
- City ID
- Level of Operation

3. Airlines

The contents are :

- Airlines ID
- Name
- Address

4. City

The contents are :

- City ID
- Name
- State or Union Territory
- Status

5. Stoppage

The contents are :

- Stoppage ID
- Arrival Time
- Departure Time
- Airport ID of Previous Stoppage
- Airport ID of Next Stoppage
- Time Flight Stays

6. Path

The contents are :

- Start City ID
- End City ID
- Number of Hops
- 1-dimensional Array of Flight-structure. There is one entry for each flight. The i -th entry contains information of flight i . A Flight-structure element stores a) Flight ID b) Airport ID, from where to catch the flight c) Airport ID, where to leave the flight d) Flight Arrival Time and e) Flight Departure Time
- Days of Operation FLAG
- Departure Time
- Total Time to travel on the path
- Total Waiting Time on the path
- Total Fare

7. Journey

The contents are :

- Start City ID
- End City ID
- Minimum Time
- Maximum Time
- Minimum Waiting Time
- Maximum Waiting Time
- Minimum Fare
- Maximum Fare

8. Time

The contents are :

- Hour
- Minute
- Date

9. Date

The contents are :

- Day
- Month

- Year
- Type of Date (sunday to saturday)

3.3 Algorithms

1. Algorithms to compute all practically feasible paths

At any instance, we have the data of all practically feasible paths for every pair of source and destination. This data depends on the flights available. Therefore, any kind of updation in the data of all flights will need the updation in the data of the paths.

Now there can be three types of updation in the data of flights:

- Insertion
- Deletion
- Modification

Insertion will introduce some new paths and the estimates of journey will change. Deletion will make some paths invalid and it also changes the estimates of journey. Modification may cause both introduction of new paths and deletion of some paths. Here we will insert new paths for insertion of new flight and delete old paths on the deletion of a flight. On modification of a flight, we first delete the paths involving that flight (before modification) and then insert new paths for the modified flight (after modification).

Algorithm - Add Paths For Flight :

Input - The flight inserted

Output - Message of success or failure

Scheme -

Begin

1. (* Find all new paths with no hops *)
 - For each pair of two stopages of the inserted flight (including the source and destination) do
 - {
 - Make a new path with no hop, from the stopage where the flight arrives first to another and store it temporary as a new path with no hop.
 - }
2. (* Find all new paths with one hop *)
 - For each old path with no hop path1 and each new path with no hop path2 do

```

    {
        if path1 is joinable with path2 or path2 is joinable with
        path1 do
            join two paths and Store it temporary as new path
            with one hop.
        (* Here for two paths path1 and path2, path1 is joinable
        with path2 if destination of path1 and source of path2 are
        the same, path1 and path2 have no common flight and
        travelling along path1-path2 does not cause a pass through a
        same plcae. *)
    }
3. (* Find all new paths with two hops *)
    For each old path with one hop path1 and each new path with no
    hop path2 do
    {
        if path1 is joinbale with path2 do
            join two paths as path1-path2 and store it as a new
            path with two hop.
        }
    For each new path with one hop path1 and each old path with no
    hop path2 do
    {
        if path1 is joinbale with path2 do
            join two paths as path1-path2 and store it as a new
            path with two hop.
        }
4. (* Find all new paths with three hops *)
    For each old path with one hop path1 and each new path with one
    hop path2 do
    {
        if path1 is joinable with path2 or if path2 is joinable with
        path1 do
            join the two paths and store this path after joining
            temporarily as new paths with three hops.
        }
5. (* Store new paths in the local database *)
    For each new path path1 do
    {
        Insert this new path path1 and update tha data of the journey
        with the source and the destination same as those of this new
        path path1.
    }
}
EndOfScheme

```

Algorithm - *Delete Paths For Flight* :

Input - The flight to be deleted

Output - Message of success or failure

Scheme -

Begin

1. for each path path1 involving the flight to be deleted do
delete path1.
2. Revise All Journeys.

EndOfScheme

2. Algorithm to compute the shortest path from source to destination, with given constraints :

Algorithm - *Shortest Path* :

Input - Source, Destination, Time Interval To Start Journey, Priorities, Class Of Journey

Output - A Shortest Path or Message of Failure

Scheme -

shortest - shortest path till moment

current - path in question at the moment

current square - sum of deviations

least square - least of all squares till moment

max time - maximum time required for the journey

min time - minimum time required for the journey

max wait - maximum waiting time required for the journey

min wait - minimum waiting time required for the journey

max fare - maximum fare required for the journey

min fare - minimum fare required for the journey

Begin

shortest = null

least square = infinity

1. for each path current with given source and destination do

{

if the departure of path is in given time interval do

{

current square = zero

importance for time = $\frac{\text{max time} - \text{total time}}{\text{max time} - \text{min time}} \times 100$

if importance is less than priority for time do

add square of difference in importance and priority
to current square

importance for waiting time = $\frac{\text{max wait} - \text{waiting time}}{\text{max wait} - \text{min wait}} \times 100$

if importance is less than priority for waiting time do

add square of difference in importance and priority
to current square

$$\text{importance for fare} = \frac{\text{max fare} - \text{total fare}}{\text{max fare} - \text{min fare}} \times 100$$

if importance is less than priority for fare do
add square of difference in importance and priority
to current square

if current square is less than least square do

{
least square = current square
shortest = current
}

}

2. if shortest is not null do
output shortest
else output message of failure

EndOfScheme

4. The User Guide

4.1 The Installation

Install the system

Create a separate directory, to install the software. Copy all files in it. Compile the programs using **the C++ Compiler**. Run the initialization programme. This programme will set the password(s) meant for the administrator and access the global database available on the Internet to create the local database consistent with it. At the end of execution of this programme, you will have the system installed.

4.2 The Maintenance

Keep the system up-to-date

You fix some time of interval (one week is recommended). Run the database updation program in the administrator mode at the time of interval, fixed by you. This programme will access the global database and update the local database. In this way, you can keep your system up-to-date.

5. Results and Conclusion

5.1 Results

The software developed so far has been used to plan travelling from Bhubaneswar to Amritsar with various constraints. The results are given here.

1. To travel on 1, Jan 1998 in minimum time

The path we got is :

From : Bhubaneswar
To : Amritsar
Total time : 20 hours and 45 minutes
Waiting time : 14 hours and 25 minutes
Total fare :

- Economy class : Rs. 16990/-
- Executive class : Rs. 25240/-
- Concession class : Rs. 12400/-

The flights are :

- IC 744 from Bhubaneswar at 15 : 30 to Calcutta at 16 : 20
- IC 765 from Calcutta at 18 : 00 to Chennai at 20 : 05
- IC 539 from Chennai at 23 : 05 to Delhi at 1 : 35
- CD 486 from Delhi at 11 : 20 to Amritsar at 12 : 15

2. To travel on 1, Jan 1998 in minimum waiting time

The path we got is :

From : Bhubaneswar
To : Amritsar
Total time : 20 hours and 45 minutes
Waiting time : 14 hours and 25 minutes
Total fare :

- Economy class : Rs. 16990/-
- Executive class : Rs. 25240/-
- Concession class : Rs. 12400/-

The flights are :

- IC 744 from Bhubaneswar at 15 : 30 to Calcutta at 16 : 20
- IC 765 from Calcutta at 18 : 00 to Chennai at 20 : 05
- IC 539 from Chennai at 23 : 05 to Delhi at 1 : 35
- CD 486 from Delhi at 11 : 20 to Amritsar at 12 : 15

3. To travel on 1, Jan 1998 paying minimum fare

The path we got is :

From : Bhubaneshwar
To : Amritsar
Total time : 20 hours and 45 minutes
Waiting time : 16 hours and 55 minutes
Total fare :

- Economy class : Rs. 9320/-
- Executive class : Rs. 13790/-
- Concession class : Rs. 6900/-

The flights are :

- IC 744 from Bhubaneshwar at 15 : 30 to Calcutta at 16 : 20
- IC 402 from Calcutta at 17 : 30 to Delhi at 19 : 35
- CD 486 from Delhi at 11 : 20 to Amritsar at 12 : 15

4. To travel on 5, Jan 1998 in minimum waiting time, paying minimum fare

The path we got is :

From : Bhubaneshwar
To : Amritsar
Total time : 22 hours and 5 minutes
Waiting time : 17 hours and 15 minutes
Total fare :

- Economy class : Rs. 11115/-
- Executive class : Rs. 16425/-
- Concession class : Rs. 8100/-

The flights are :

- IC 706 from Bhubaneshwar at 14 : 10 to Guwahati at 14 : 50
- IC 704 from Guwahati at 15 : 30 to Calcutta at 16 : 40
- IC 402 from Calcutta at 17 : 30 to Delhi at 19 : 35
- CD 486 from Delhi at 11 : 20 to Amritsar at 12 : 15

5.2 Conclusion

Here we have developed a software, which can help a common man in planning a travelling by air in India. In this project, we have assumed that a traveller want to travel from one place to another. We can upgrade this software so that it can help to plan a tour (visiting more than one place) through airlines within India.

References

1. "Software Engineering - A Practitioner's Approach" by Roger S. Pressman. The McGraw Hill Companies Inc., New Delhi.
2. "An Integrated Approach to Software Engineering" by Pankaj Jalote. Narosa Publishing House, New Delhi.
3. "Database System Concepts" by Korth and silberschatz. The McGraw Hill Companies Inc., New Delhi.