# RAILWAY ENQUIRY AND RESERVATION SYSTEM

Y6470 – SIDDHARTH SHANKER
Y6377 – RASHISH TANDON
Y6136 – BANDA GIRISH KUMAR

CS - 315

*Project Guided By:*
*Prof. Arnab Bhattacharya*
*Presented:*
*2008 – 09 Semester - II*

# INDEX

## ➢ INTRODUCTION

This project aims at development of an Online Railway Reservation Utility which facilitates the Railway customers to manage their reservations online, and the Railway administrators to modify the backend databases in a User-Friendly manner.

The Customers are required to register on the server for getting access to the database and query result retrieval. Upon registration, each user has an account which is essentially the 'view level' for the customer. The account contains comprehensive information of the user entered during registration and permits the customer to get access to his past reservations, enquire about travel fare and availability of seats, make afresh reservations, update his account details, etc.

The Railway Administrator is the second party in the transactions. The administrator is required to login using a master password, once authenticated as an administrator, one has access and right of modification to all the information stored in the database at the server. This includes the account information of the customers, attributes and statistics of stations, description of the train stoppages and physical description of coaches, all the reservations that have been made, etc. The railway administrator has the right to modify any information stored at the server database.

## MOTIVATION

This project is dedicated to

- model the existing Railway/(other) reservation systems
- provide a comprehensive set of features to enhance their operational limits
- evaluate their performance in different scenarios
- suggest modifications for greater efficiency

## SCOPE AND FEASIBILITY

Implementations of this project idea are in industrial use in the form of 'www.irctc.co.in', etc. Hence, this can be used for suggesting improvements in design, performance and greater usability.

Apart from the industrial applications mentioned above, it is a research oriented project as well, the task of performance evaluation of different database designs, for efficiency, is in this spirit.

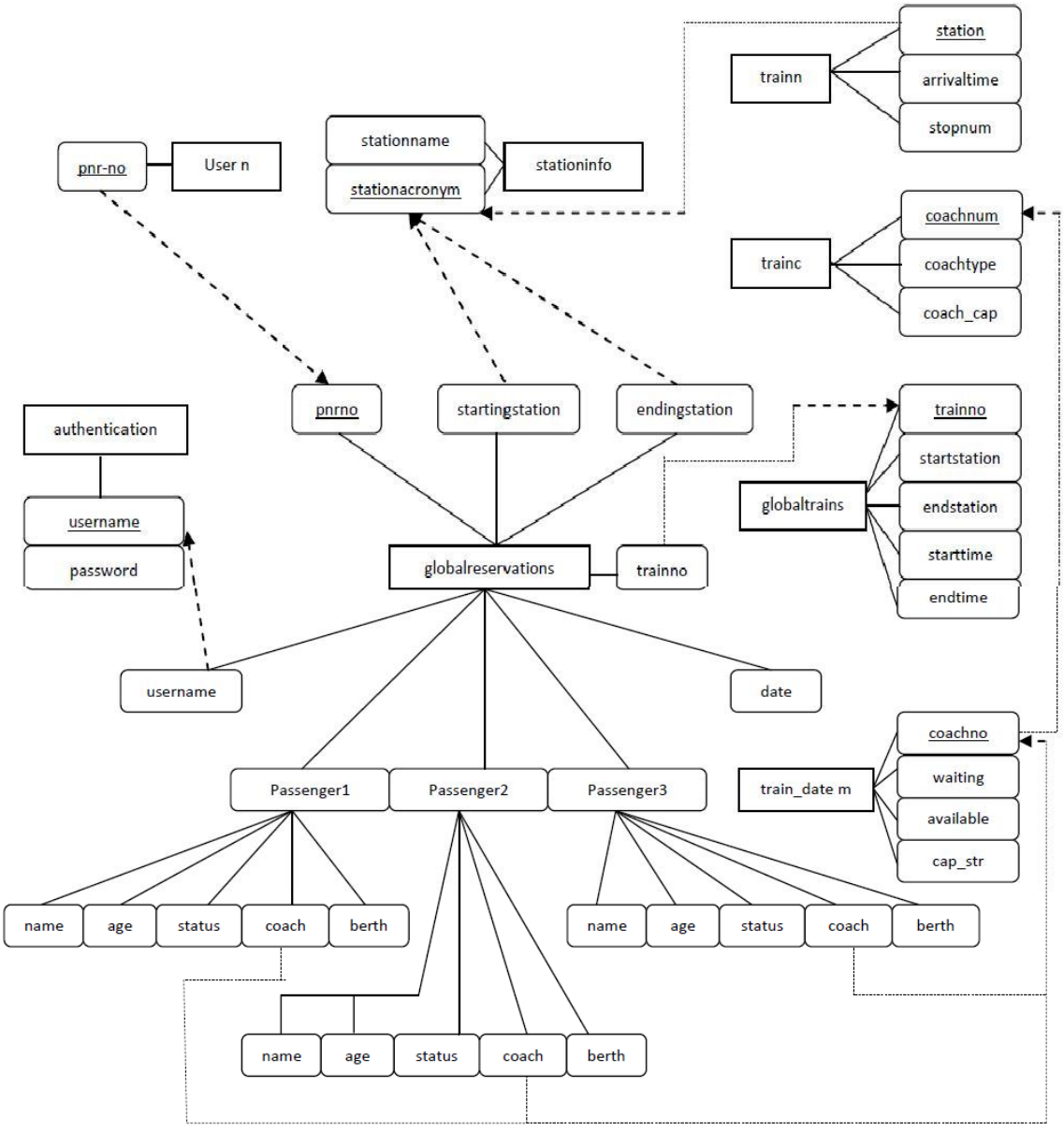
## FUNCTIONALITY

The Customer and the Railway Administrator are the two parties which interact with the database, who have different 'view level schemas' to the database information.

- Customer Services
    i. Create an account by registering, modify account details, deregister from the services
    ii. Make afresh multi passenger reservations, the customers are provided to choose their berths/reservation spots rather than being randomly allocated positions
    iii. View , modify or cancel past reservations
    iv. Customers are provided with different reservation status, just as in real life systems
    v. Consumers are informed, through emails, about updates in the reservations and trains
    vi. Consumers are informed about the various seasonal offers and discounts.

- Administrator Services
    i. Add new train services or update the existing train services, e.g. modifying their stopping stations, stoppage times, tariffs, etc.
    ii. Add or update the physical description of trains, like number of coaches, type of coaches, number of berths, etc.
    iii. Update information about addition of railway stations, add new railway stations, drop existing railway stations, etc.
    iv. Access and modify customer accounts or customer reservations

- **REFERENTIAL DIAGRAM**



## ○ DATA DICTIONARY

- <User> e.g.(ssidha, rashish)

| Attribute | Type | Constraints | References |
|---|---|---|---|
| Pnrno | Integer | Primary key | Pnrno of globalreservations |

- <train>n e.g.(2300n, 2200n)

| Attribute | Type | Constraints | References |
|---|---|---|---|
| Station | Varchar(4) | Primary key | Stationacronym of stationinfo |
| Arrivaltime | Time | - | - |
| Stopnum | Integer | Candidate key | - |

- <train>c e.g.(2300n, 2200n)

| Attribute | Type | Constraints | References |
|---|---|---|---|
| Coachnum | integer | Primary key | Stationacronym of stationinfo |
| coachtype | varchar(4) | - | - |
| coachcap | Integer | - | - |

- Authentication

| Attribute | Type | Constraints | References |
|---|---|---|---|
| Username | Varchar(4) | Primary key | - |
| Password | varchar(4) | Not NULL | - |

- globalreservations

| Attribute | Type | Constraints | References |
|---|---|---|---|
| pnrno | integer | Primary key | - |
| startingstation | varchar(4) | Not NULL | Stationacronym of stationinfo |
| endingstation | varchar(4) | Not NULL | Stationacronym of stationinfo |
| trainno | integer | Not NULL | trainno of globaltrains |
| date | integer | Not NULL | - |
| username | varchar(20) | Not NULL | Username of authentication |
| pass1_name | varchar(20) | Not NULL | - |
| pass1_age | integer | Not NULL | - |
| pass1_coachnum | integer | Not NULL | Coachnum of <trainno>c, <train>_<date> |
| pass1_waitnum | integer | Not NULL | - |
| pass1_berthnum | integer | Not NULL | - |
| pass1_status | integer | Not NULL | - |
| pass2_name | varchar(20) | - | - |
| pass2_age | integer | - | - |
| pass2_coachnum | integer | - | Coachnum of <trainno>c, <train>_<date> |
| pass2_waitnum | integer | - | - |
| pass2_berthnum | integer | - | - |
| pass2_status | integer | - | - |
| pass3_name | varchar(20) | - | - |
| pass3_age | integer | - | - |
| pass3_coachnum | integer | - | Coachnum of <trainno>c, <train>_<date> |
| pass3_waitnum | integer | - | - |

| | | | |
|---|---|---|---|
| pass3_berthnum | integer | - | - |
| pass3_status | integer | - | - |

- globaltrains

| Attribute | Type | Constraints | References |
|---|---|---|---|
| trainno | integer | Primary key | - |
| startstation | varchar(4) | Not NULL | Stationacronym of stationinfo |
| endstation | varchar(4) | Not NULL | Stationacronym of stationinfo |
| starttime | time | Not NULL | Arrivaltime of <trainno>n |
| endtime | time | Not NULL | Arrivaltime of <trainno>n |

- stationinfo

| Attribute | Type | Constraints | References |
|---|---|---|---|
| Stationacronym | varchar(4) | Primary key | - |
| stationname | varchar(20) | - | - |

- <trainno>_<date> e.g. (2300_101, 2400_102)

| Attribute | Type | Constraints | References |
|---|---|---|---|
| coachno | Integer | Primary key | Coachnum of <trainno>c |
| available | integer | Not NULL | - |
| cap_str | varchar(100) | Not NULL | - |
| waiting | integer | - | - |

Notes:

i. For each customer / user, there is a file named <username> of the user which stores the PNR number of all recent the user has made. The details of the reservation can be obtained by performing a “natural join” with the globalreservations table. The table is updated on new reservation being made and an existing reservation being cancelled.

ii. For each train, there are 2 tables which describe the train,
   a. <train>n – This stores the stations at which a train stops, the arrival times at the stations and the stop numbers of the stations. This is required to provide the facility to a customer of being able to make reservations for intermediate stops and need not book the ticket for the whole journey. Upon a request for showing trains to book a journey, a query is made to each <train>n table

to match the stations and the stoppage number ensures that the train arrives at the starting station before the ending station. This can be updated only by the administrator. A new table of such type is created and the records are inserted upon the request by the administrator to announce a new train route. To ensure the integrity of the table the administrator is provided with a drop – down box for choosing stations to protect from the accidental error of entering non-existent stations.

b. <train>c – This stores the physical description of the coach types available in the train. It stores the coach number of a coach, the respective coach capacity and the coach type. This is to provide the customers with the flexibility of choosing the coach type. This also enables the administrator to make new types of coaches available to the customer for bookings. When the customer is provided with a list of trains that serve a particular station pair (route), then he is provided with the facility of viewing the coach types of various classes in the train which he selected from the list. This can be updated only by the administrator. A new table of such type is created and the records are inserted upon the request by the administrator to announce a new train route.

iii. Each user is provided with an account, which is used for session management, the authentication table stores the account information for each user. The customer / user is provided to view and make reservations only when he enters a legitimate username and password pair. Upon a successful authentication, a php session is initiated. This table is updated upon registration of a new customer, deregistration of an existing customer or by means of a change in password request by an existing user.

iv. The whole database is centred about the table ‘globalreservations’, this table stores the complete details of all recent reservations. It stores the boarding and de-boarding stations, arrival times, date of journey, passenger list, the coach and berths allotted, etc. To ensure the integrity of this table, the users are provided with drop down boxes to choose stations and date. This table is modified upon reservation, cancellation, update in reservation of existing reservations, which may be explicit in the form of changing the passenger details, or implicit due to change in reservation status from ‘wait list’ to ‘confirmed’. This table is queried upon after being joined with the <user> table when a customer wishes to view his recent reservations. This is undoubtedly the central and the key link relating data entities in the database.

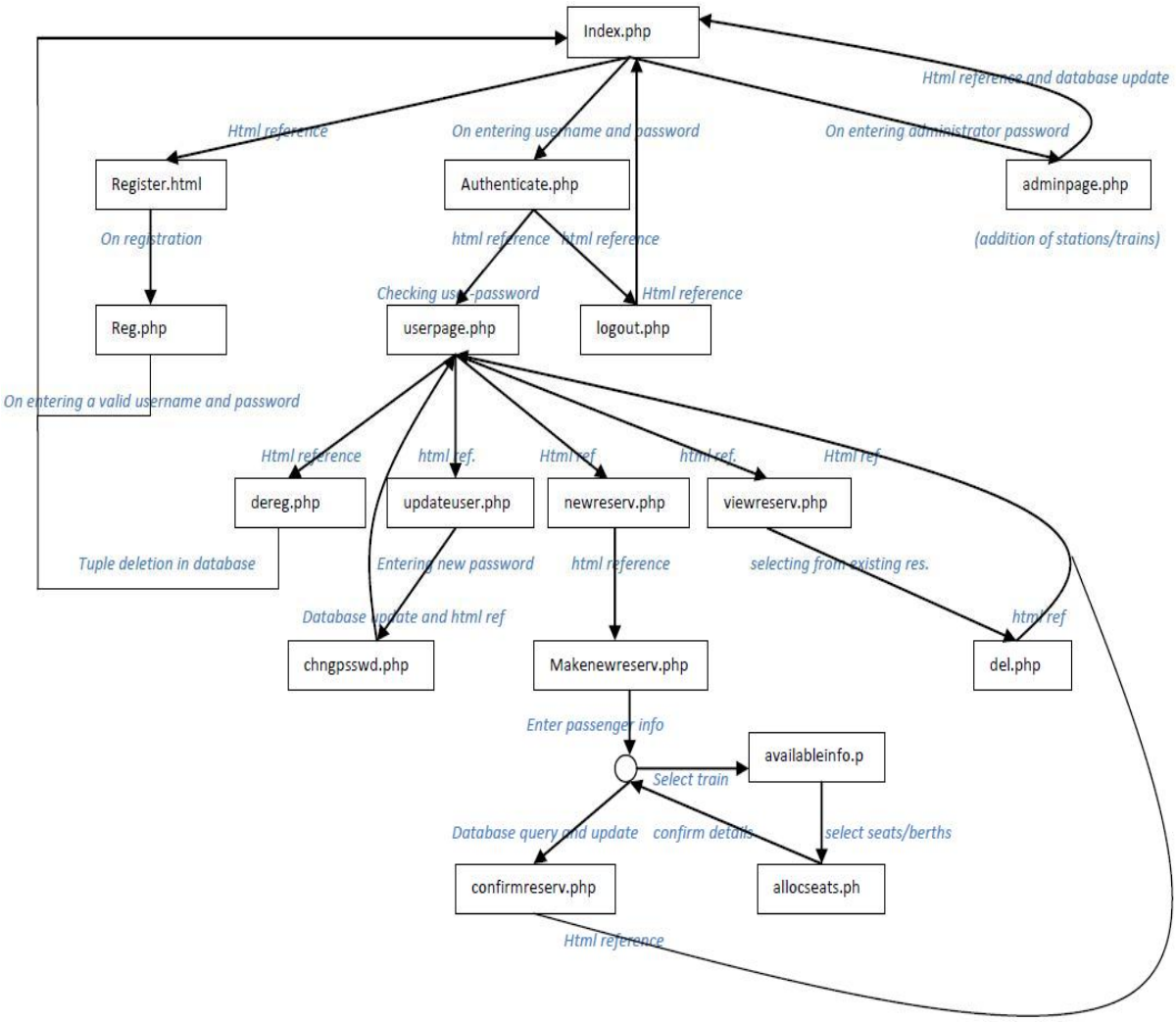
v. For making reservations, the table globaltrains consists of the identification details of all existing trains and their routes. In the query to make afresh reservation, this table provides the names of the tables whose station list needs to be searched for finding the in-order station pairs of the journey.

vi. Just as for trains, station details are stored in the table named ‘stationinfo’. This is used when the administrator tries to input new trains as a service to the customers. This constraints the administrator to choose, the start and ending stations, from a drop down menu and preventing accidental input of non existing stations.

vii. Each train and date pair for a recent date, have a table <trainno>_<date> describing the current reservations that have been made in the train <trainno> on date <date>. For each coach, a string of length equal to the capacity of the coach denotes the status of the reservation in it. The string consists of ‘o’ and ‘e’, which essentially flag whether the berth / seat is occupied or empty. New instances are added automatically upon change of date and past tables are dropped. The attribute waiting is stored to store the current

count of ‘wait listed’ reservations that have been booked for a particular coach type for the train on that date.

- **DATA FLOW CHART**



- ## SESSION – MANAGEMENT

  The data and the files present at the server are password locked. Hence, to gain access to them the user has to pass through the authentication process at ‘index.php’ / ‘adminauthenticate.php’, wherein a php session is initiated and the relevant session variables get initiated. If a client tries to access the locked files without starting a session, it is denied access and redirected to request for starting a session.

## ➢ PERFORMANCE ANALYSIS

In this section, we evaluate the performance depending on the following variables to provide generic expressions rather than measure the times on a particular server-client-network pair

- Database design in terms of the Normal forms the schemas conform
- Number of joins, selections, projections, etc. for various forms of queries
- Average access time over the internet

Terminology

a) $N_{<x>}$ : denotes the number of records in table <x>
b) $A_{<v,x>}$ : denotes the number of different values of attribute <v> in table <x>. This is analogous to selectivity of <v> in <x>
c) RTT : denotes the average 'round trip time' over the network chosen

We make the following assumptions for estimating our reservation engine performance

a) The time for equi-join on primary key attributes is proportional to the number of tuples in the result. This assumption is justified as 1 record of a each table will contribute to only 1 output of the result
b) The time for equi-join on non – key attributes on relation <x> and <y> is proportional to $N_{<x>} * N_{<y>}$. This serves as a bound
c) The time for selection on primary key in table <x> is proportional to $\log(N_{<x>})$. This is reasonable as we need to scan a path along the B-Tree
d) The time for selection on equality of non-key attribute <v> in table <x> is proportional to $(N_{<x>}/D_{<v,x>})$

We evaluate the service time of frequent operations i.e. reservation and cancellation. The administrator functions are assumed to be very rare compared to user / customer activities

- Reservation

| Activity | Operation | Time |
|---|---|---|
| **Authentication** | Selection on authentication | $N_{authentication}$ + RTT |
| **Following HTML link** | - | RTT |
| **Enter destinations info** | Selection on stationinfo | $N_{stationinfo}$ + RTT |
| **Selecting Train** | Search in all <train> after selection from globaltrains | $N_{globaltrains}$ + $N_{<train>}$*$N_{globaltrains}$ + RTT (*) |
| **Selecting Seats** | Selection from <train>_<date> | $N_{<train>\_<date>}$ + RTT |
| **Database update** | Update <user>, globalreservations, <train>_<date> | $N_{<user>}$ + $N_{globalreservations}$ + $N_{<train>\_<date>}$ + RTT |

*Bottleneck operation highlighted

- Cancellation

| Activity | Operation | Time |
|---|---|---|
| **Authentication** | Selection on authentication | $N_{authentication}$ + RTT |
| **Following HTML link** | - | RTT |
| **Selecting from reservations** | Natural join on globalreservations and <user> | $N_{globalreservation}$ / $A_{<user>}$ + RTT |
| **Selecting other tuples to be updated** | Search and update tuples | $N_{globaltrains}$ + RTT (*) $N_{globaltrain}$/$A_{trainno,date,<train>\_<date>}$ + |
| **Deallocating Seats** | Selection from <train>_<date> | $N_{<train>\_<date>}$ + RTT |
| **Database update** | Update <user>, globalreservations, <train>_<date> | $N_{<user>}$ + $N_{globalreservations}$ + $N_{<train>\_<date>}$ |

*Bottleneck operation highlighted

## ➢ EXTERNAL INTERFACE REQUIRMENTS

The database engine can be embedded on a platform using

i. An active mySQL server
ii. A browser which acts as a client
iii. An Apache HTTP server

All the mentioned resources are free source and can be obtained easily from the internet.

## ➢ NOTES AND SUGGESTED FUTURE WORK

i. We have provided the ability where at max 3 people can be booked reservations for, this may not be desirable.
ii. We can extend it to provide multi-train service for connecting 2 stations, i.e. if there is no direct train between 2 stations, then we may use more than 1 train to book our reservations.
iii. The analysis assumes conservative and uniform spread of values, this may not hold and more sophisticated tools for estimation should be used.
iv. Greater information about the user can be stored and the user may be updated about changes in reservation status upon cancellations, via email

## ➢ ACKNOWLEDGEMENT

We bestow our hearted appreciation and gratefulness to the efforts made by our Instructor In-charge: Prof. Arnab Bhattacharya, in terms of the Project Idea, instruction and the opportunity, without which our efforts would have remained astray.