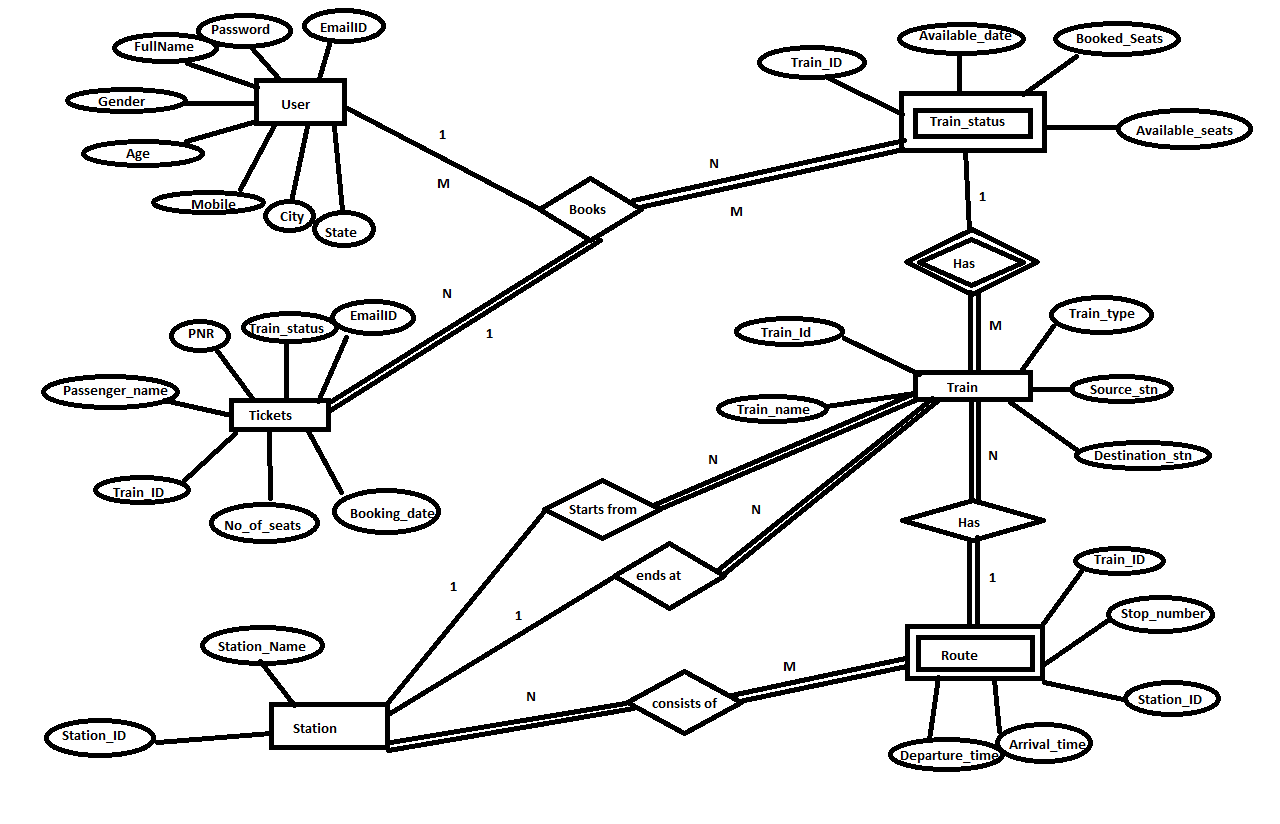
REPORT

Railway Reservation System

Logical Design:

ER DIAGRAM

****

**RELATIONAL SCHEMA**

|  |  |  |
| --- | --- | --- |
| S.no | Entity Type | Relation |
| 1 | User | User(email\_id,Password,Full Name ,Gender ,age , Mobile,City,State) |
| 2 | Tickets | Tickets(PNR, Passenger Name, Train id, No Of seats, Train\_status, Booking date) |
| 3 | Train | Train(Train id, Train\_name, Source\_stn, destination\_stn, Souce\_id, destination\_id) |
| 4 | Station | Station(Staion\_id, Station\_name) |

**Step 1: Mapping Of Strong Entity Type**

**Step 2:**

**Mapping Of Weak Entity Type**

|  |  |  |
| --- | --- | --- |
| S.no | Entity Type | Relation |
| 1 | Route | Route(Train\_Id,Stop\_number,Station\_id,Arrival\_time,departure\_time) |
| 2 | Train\_status | Train\_status(Train\_id,Available\_date,booked\_seats,Available\_seats) |

**Step 3: 1:1 Relationship –There is no 1:1 relations**

|  |  |  |  |
| --- | --- | --- | --- |
| Relation | Entities | Approach | Relation |
| Starts from | Train, station | Foreign Key | Train(Train id, Train\_name, Source\_stn, destination\_stn, Souce\_id, destination\_id) |
| Ends at | Train, Station | Foreign Key | Train(Train id, Train\_name, Source\_stn, destination\_stn, Souce\_id, destination\_id) |

**Step4: Mapping Of 1 : M Relationship:**

**Step5: Mapping Of M:N Relationship:**

|  |  |  |  |
| --- | --- | --- | --- |
| Relation | Entities | Approach | Relation |
| Consists of | Station, Route | Cross-Reference | Route\_has\_station(Train\_id,station\_id,stop\_number) |

**Step6: Multivalued Attributes:** There Are no multivalued attributes**.**

|  |  |  |  |
| --- | --- | --- | --- |
| Relation | Entities | Approach | Relation |
| Books | Users, Tickets,train\_status | Cross-reference | Tickets(PNR, Passenger Name, Train id, No Of seats, Train\_status, Booking date) |

**Step7: Mapping Of n-array Relationship**

**Relational Schema:**

**USERS**

Email Id Full name Password Age Gender Mobile City State

Ed

TRAIN

Train Id Train-Name Train-type Source Destination

STATION

Station Id Station Name

TICKETS

PNR Passenger Name No Of seats Booking date Train Id Email Id

ROUTE

Train Id Stop\_number Station Id Arrival time Departure time

TRAIN STATUS

Train Id Available Date booked Seats Available Seats

CONSISTS OF

Train Id Station Id Stop Number

**FIRST NORMAL FORM**

In train, days\_Available is a multivalued Attribues. So we are decomposing Train\_id and days\_available into a separate table.

**2nd : Normal** Form Is not there

**3rd Normal Form**

In train Relation Source\_id and source\_stn are non-key attributes and there is transitive dependency between them, so there are not in third normal form.

In train Relation destionation\_id and destination\_stn are non-key attributes and there is transitive dependency between them, so there are not in third normal form.

So we are decomposing those attributes.

**Physical Design**

**Railway Database:**

Structure of customer:

|  |  |
| --- | --- |
| Email\_id | Char[30] |
| password | Char[15] |
| Full\_name | Char[30] |
| Gender | Char[8] |
| Age | int[11] |
| Mobile | Char[14] |
| City | Char[20] |
| State | Char[25] |

Total size of each record(tupple) = 164 Bytes

*Computation of the Blocking Factor for the table using the standard block size of 512 bytes.*

Blocking factor = 512/164 = 3 Records per block.

Number of file blocks = 25,00,000/3 = 8,33,334 blocks.

**Storage Requirements:**

Number of bytes used per block= 3\*164=492

Percentage wastage = (512-492/512) x 100 = (20/512) x 100 = 3.90625%

Hence unspanned file type is used

Each record effectively occupies (512/3) = 170.66bytes.

Overhead of 0.66 bytes per record.

**Access Method:**

1. Binary search on this data file = log2(8,33,334) = 20 block(if it is orderd) but insertion and update are difficult.
2. Average linear search=(no of blocks)/2=416667 block accesses.

To decrease the no of accesses, we use efficient file structure called indexing.

**Why indexing ??**

* **By using indexing we can reduce the no of block accesses.**

1. **Primary Indexing:**

Length of each index record = 30 bytes (Primary Key) + 5 bytes (Block Pointer) = 35bytes

Index blocking factor = 512/35= 14

Total number of entries in index file = Number of blocks = 8,33,334

Therefore, number of blocks required 8,33,334 /14 = 59524 blocks

To perform binary search on index file log2(59524) = 16 block accesses

To search for a record we need 16+1 =17 block accesses ( to access the data file block containing record)

1. **Secondary Indexing:**

Clustering index on state( after ordering)

Size of state = 25 bytes

Size of index entry = 25 + 5 = 30bytes

Blocking Factor = 512/30= 17 index records per block

Total number of index records = 29

Number of blocks needed for index file = 2

Percentage wastage = 34-29 =5 => (5/17) x 100 = 29.4%

1. **Second level Indexing:**

Number of indexes = = 8,33,334/14 = 59524blocks

Blocking factor = 512/35 = 14 records per block

Hence, number of blocks required at second level = 4252 blocks.

**B – Trees:**

P x 5 + (P-1) x 30+ (P-1) x 5 < 512

40P-35 < 512

40P < 547

P < 13.675

P = 13

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Number of Nodes | Number of entries | Number of pointers |
| Root | 1 | 12 | 13 |
| Level 1 | 13 | 156 | 168 |
| Level 2 | 168 | 2016 | 2184 |

Number of blocks needed at level n = 1000/12 = 84 blocks

Number of blocks needed at level n-1 = 84/12 = 7 blocks

Total number of blocks = 84 + 7 = 91 blocks

**B+ Trees:**

P x 5 + (P-1) x 30<= 512

35P-30 <= 512

35P <= 542

P <=15.48

P=15

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Number of Nodes | Number of entries | Number of pointers |
| Root | 1 | 14 | 15 |
| Level 1 | 15 | 210 | 225 |

Number of blocks needed at level n = 1000/14 = 72 blocks

Number of blocks needed at level n-1 = 72/14 = 6 blocks

Total number of blocks = 72+6 = 78 blocks.

**DATABASE SIZE:**

Size of each record in customer table =164 bytes.

Total no of records=25,00,000.

So,

size of database = total no of records x size of each record.

=2500000 x 164 = 410000000 bytes

=391.00647MB.

**QUERY COST:**

Block access time = seek time + Rotational delay + block transfer time

+ overhead +queing delay

Assume,

Seek time =20ms, rotational delay=1ms, block-transfer time = 1ms, overhead=0ms , queing delay=0ms

Then block transfer time =seek time + rotation delay +block transfer time

=20ms+1ms+1ms

=22ms

For one block accesses it takes 22ms.

If there are no buffers,then

Query retrival time for scaning

**Heap files:**

no of block access x block aceess time

= 416667 x 22

=9166674ms.

**Ordered files:**

No of block access x block access time

=20 x 22

= 440

**Indexed files:**

**Primary index:**

no of block access x block access time

=17 x 22

=374