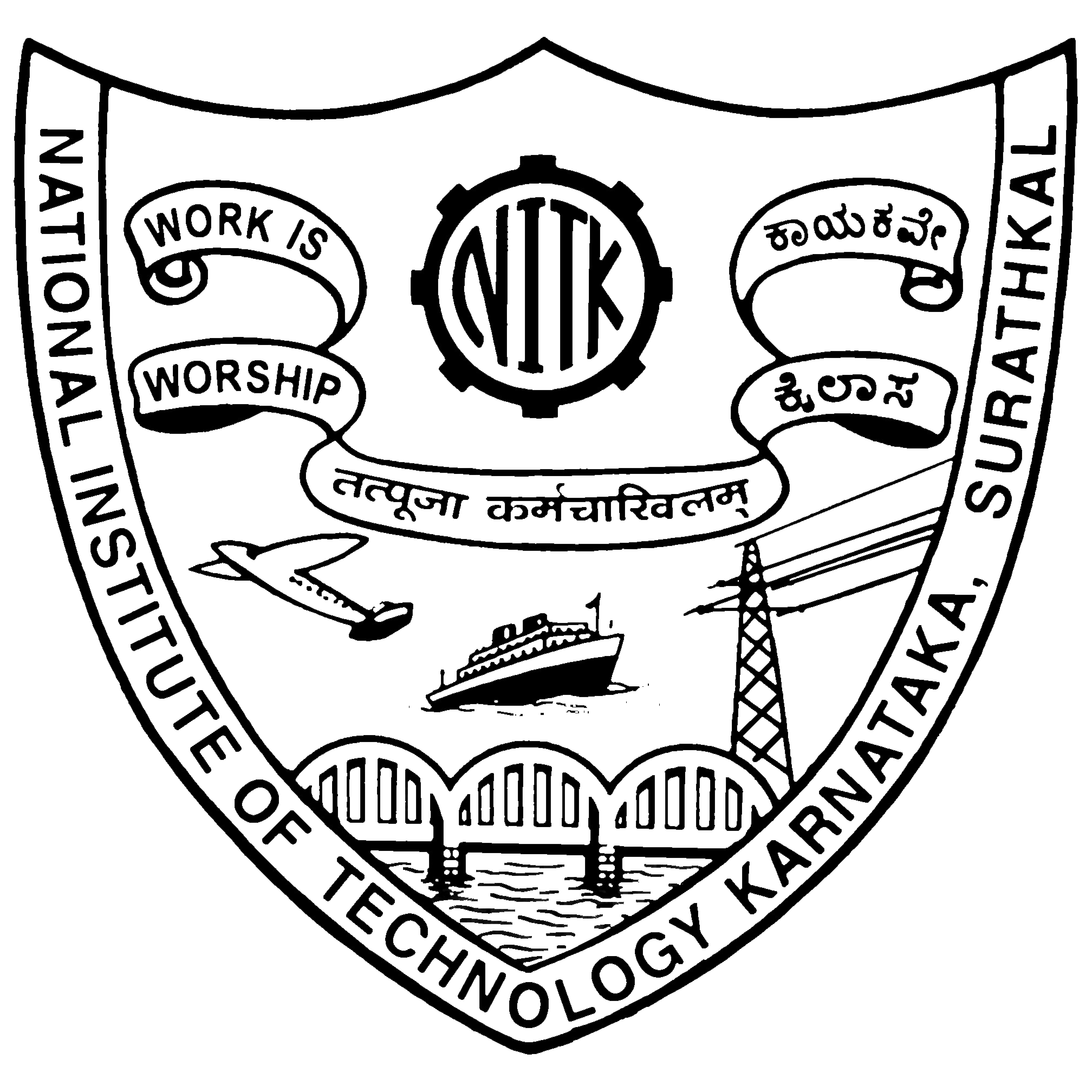
*NITK, Surathkal*

*Department of Information Technology*



**Advanced Database Systems**

**Assignment 1**

Book Your Trip – The Air Way

**Name**: Sachin

**Roll No**: 17IT019

**TABLE OF CONTENTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Problem Description |  |  |  |  | 3 |
| Entity Relationship Diagram |  |  |  |  | 5 |
| Relational Schema |  |  |  |  | 6 |
| Normalization |  |  |  |  | 7 |
| Fragmentation |  |  |  |  | 8 |
| Horizontal Fragmentation |  |  |  |  | 9 |
| Vertical Fragmentation |  |  |  |  | 10 |
| Physical Design |  |  |  |  | 28 |
| Replication And Allocation |  |  |  |  | 29 |
| Disk Parameter |  |  |  |  | 30 |
| Indexing |  |  |  |  | 34 |

**Problem Statement:**

**Focus of the Problem:**

The ”Book Your Trip – The Air Way“ database project is an attempt to stimulate the basic concepts of trip booking system on a “distributed environment”. Basically, the system enables the user to search and book flights between two cities, and to search and book cab from their location to nearest airport or vice versa and allows user to search and book hotels or any home stays for accommodation during their entire trip.

**Data Flow and Control Flow:**

To book a flight, user has to create an account in the system to avail the facilities by providing details such as first name, last name, Aadhaar number, email ID, phone number, gender, age, permanent address. Once the account is created, then user can search the flights by providing traveling details such as departure city, arrival city, date of travel, time, number of passengers and travel class. Based on user data, system will show all the flights available, then user has to select flight based on their convenience by looking at airlines company, fare and direct flight or via some other airport. If there are no seats available, user has to select another flight and if there are any other travelers, then user needs to enter their details such as first name, last name, gender and age, after that system will redirect user to payment page where user needs to provide the payment details such as debit/credit card number, cardholder name and password, then a pop-up message will appear about the payment successful or not. If payment is unsuccessful, then user needs to do payment once again.

While booking a flight, user is asked to avail the additional services provided by the system such as cab service, hotel service and home stay then user is redirected to particular page.

As user wants to book a cab, system takes user details such as first name, last name, phone number which are provided while booking the flight or hotel or home stay and user needs to enter some additional details such as pickup point and drop point and pick up time, then the system will book the cab for the user and a message is sent to user on a registered phone number with cab number, driver name and driver contact number after that user can do payment after reaching the drop point.

As user wants to book a hotel, system takes user details such as first name, last name, Aadhaar number, phone number, permanent address which are provided while booking the flight and user needs to enter some additional  accommodation details such as city or area where the user is going to stay, check in, checkout date and time and number of guests. Based on user data, system will show all the available hotels, then user has to select a hotel according to their convenience by looking at nearest hotel, fare and reviews. After that user can book any hotel and can do payment at the time of check in or checkout.

As user wants to book a homestay, system takes user details such as first name, last name, Aadhaar number, phone number, permanent address which are provided while booking the flight and user needs to enter some additional accommodation details such as city or area where the user is going to stay, check in, checkout date and time and number of guests and number of rooms required in a home stay. Based on user data, system will show all the available home stays, then user needs to select a home stay according to their convenience by looking at nearest home stay, fare and reviews. After that user can book  a homestay and can do payment at the time of check in or checkout.

**Queries:**

1. Retrieve the flights details between two cities.
2. Book a cab from user location to some airport to board a flight to some city.
3. Booking of hotel/ home stay for some user in arrival city.
4. Booking of cab from some airport to user booked hotel/home stay in arrival city.
5. Booking of cab for a user to airport and booking a hotel/home stay in arrival city.
6. Update flight/cab/hotel/home stay details of some user.
7. Cancel flight/cab/hotel/home stay of some user.

**Constraints:**

* To update any changes in trip/hotel/home stay details, user has to update it before 12 hours of departure time, after that no updates will be allowed.
* To cancel a cab/flight ticket/hotel/home stay, user has to cancel it before 12 hours of departure time and refund policy will be decided by airlines.
* While boarding the flight, cab should be booked such that passenger should reach the airport at least one hour before the boarding time.
* On arrival side of the flight, the cab/hotel/home stay should be booked after arrival time of the flight.

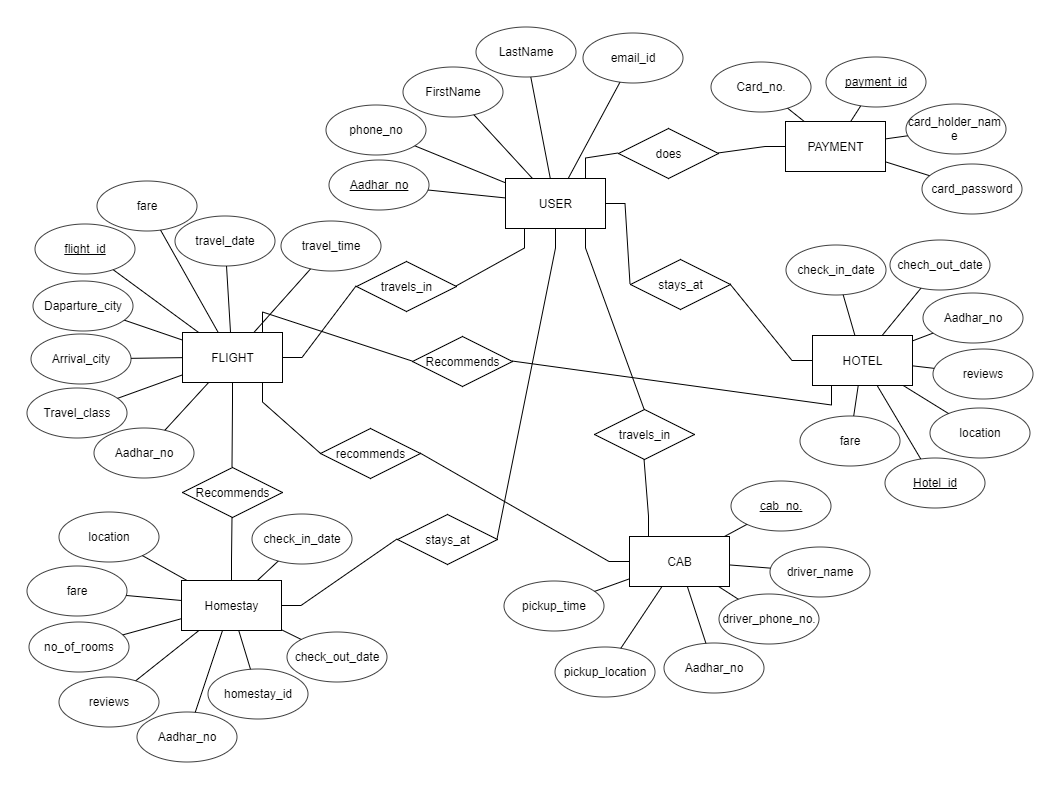
**Number of sights/locations:**

* User site
* Travelling site
* Accommodation site

**Entity Sets:**

1. User (Aadhar\_no, FirstName, LastName , email\_id, phone\_no., )
2. Flight (Flight\_id, Departure\_city, Arrival\_city, fare, travel\_class, Aadhar\_no , boarding \_date, boarding \_time)
3. Cab (Cab\_no, driver\_name, driver\_phone\_no, Aadhar\_no, pickup\_location, pickup\_time)
4. Hotel (Hotel\_id, Hotel\_name, fare, Location, Aadhar\_no, check\_in\_date, check \_out\_date)
5. Homestay (homestay\_id, Homestay\_name, location, fare, no\_of\_rooms, Aadhar\_no, check\_in\_date, check \_out\_date)
6. Payment (payment\_id, card\_no, card\_holder\_name, card\_password)

**E-R Diagram**

****

**SCHEMA**

USER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aadhar\_no | FirstName | LastName | email\_id | phone\_no |

FLIGHT

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Flight\_id | Departure\_city | Arrival\_city | fare | Travel\_class | Aadhar\_no | boarding \_date | boarding \_time |

CAB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cab\_no | driver\_name | driver\_phone\_no | Aadhar\_no | Pickup\_location | Pickup\_time |

HOTEL

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hotel\_id | Hotel\_name | fare | Location | Aadhar\_no | Check\_in\_date | Check\_out\_date |

HOMESTAY

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Homestay\_no | Homestay\_name | fare | Location | Aadhar\_no | Check\_in\_date | Check\_out\_date | no\_of\_rooms |

PAYMENT

|  |  |  |  |
| --- | --- | --- | --- |
| payment\_id | card\_no | card\_holder\_name | card\_password |

**NORMALIZATION**

Normalization is the process of efficiently organizing data in a database. There are two goals of normalization process: eliminating redundant data and ensuring data dependencies make sense. Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

The above Global Schemas are normalized.

**1st Normal Form:**

All the relations specified above are in 1st Normal Form since none of the relations above have multiple value in the same row for a particular field ie all the values are atomic in nature.

**2nd Normal Form**:

All the relations are in 2nd Normal Form as the relations are already in 1st NF and there are no partial functional dependencies on the non prime attributes on the primary key and also the candidate keys.

**3rd Normal Form and BCNF:**

All the relations are in 3rd NF and BCNF, since they are in 2NF and every non prime attributes of all the above relations are non transitively dependent on every candidate key of the respective relation.

**FRAGMENTATION**

Fragmentation is the process of dividing a relation into two or more sub-relations to facilitate query execution. There are two types:

1. Horizontal Fragmentation

2. Vertical Fragmentation

Horizontal Fragmentation partitions a relation along its Tuples while Vertical Fragmentation partitions a relation along its Attributes.

Both Horizontal and Vertical Fragmentation are based on the queries.

The Queries given in the problem description are rewritten in SQL below:

**Queries**:

1. **Retrieve the flights details between two cities.**

SELECT flight\_id, Departure\_city , Arrival\_city

FROM Flight

WHERE Departure\_city =’ MANGALURU’ AND Arrival\_city =’KOLKATTA’

1. **Book a cab from user location to some airport to board a flight to some city.**

SELECT \*

FROM Cab

WHERE pickup\_time IN( SELECT boarding\_time, Departure\_city, Arrival\_city

FROM Flight

WHERE Cab.pickup\_time < Flight.boarding\_time AND

Departure\_city= ‘BANGALORE’ AND

Arrival\_city= ’MUMBAI’ )

1. **Booking of hotel for some user in arrival city.**

SELECT Hotel\_id, Location

FROM Hotel

WHERE Location IN ( SELECT

FROM Flight

WHERE Hotel.Location = Flight.Arrival\_city AND

Aadhar\_no =’0000000001’ )

1. **Booking of cab from some airport to user booked home stay in arrival city.**

SELECT Cab\_no, pickup\_location

FROM Cab

WHERE pickup\_location IN ( SELECT

FROM Flight

WHERE Cab.pickup\_location = Flight.Arrival\_city AND SELECT Homestay\_id

FROM Homestay

WHERE Homestay\_id = ’111’)

1. **Booking of cab for a user to airport and booking a hotel in arrival city.**

SELECT cab\_no, Location, Aadhar\_no

FROM Cab,Flight, User

WHERE Cab.Location= ‘MANGALURU’ AND Flight.Aadhar\_no= User.Aadhar\_no

AND SELECT Hotel\_id, Hotel\_name, Location, Arrival\_city

FROM Hotel, Flight

WHERE Flight.Arrival\_city = Hotel.Loation

1. **Update flight details of some user.**

SELECT boarding\_date, fare, Aadhar\_no

FROM Flight, User

WHERE Flight.Aadhar\_no = User.Aadhar\_no AND UPDATE boarding\_date =’01/01/2018’ AND payment\_id =’000000222’

1. **Cancel home stay of some user.**

DELETE

FROM Homestay, User

WHERE Homestay.Aadhar\_no =User.Aadhar\_no AND payment\_id =’000000123’

**HORIZONTAL FRAGMENTATION**

The predicates that are formed from the following query are-

1. Pickup\_time =’changes’

Since pickup\_time changes every once in a while, so there are lot of values. Hence we can’t do horizontal fragmentation for all values of it.

Therefore, we can directly proceed for vertical fragmentation

**VERTICAL FRAGMENTATION**

Inputs to the Vertical Fragmentation step are the Frequency Matrix, the Usage Matrix and the Attribute Affinity Matrix.

1. Frequency matrix specifies the frequency measure of each query from each site.

2. Usage Matrix specifies the attributes of a relation that a query accesses.

3. Attribute Affinity Matrix specifies the affinity measure of each pair of attributes.

The sites mentioned in the above problem description are as below:

S1=User Site

S2=Travelling Site

S3=Accommodation Site

For queries given in the problem description, assume the frequency matrix is given below:

**Frequency Matrix:**

Assume frequency matrix as:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sites**  **Queries** | **S1** | **S2** | **S3** | **Total Queries** |
| **Q1** | 0 | 50 | 0 | 50 |
| **Q2** | 0 | 40 | 0 | 40 |
| **Q3** | 0 | 20 | 30 | 50 |
| **Q4** | 0 | 20 | 0 | 20 |
| **Q5** | 20 | 0 | 0 | 20 |
| **Q6** | 20 | 20 | 0 | 40 |
| **Q7** | 20 | 0 | 20 | 40 |

Now first we would be considering the relation User.

**RELATION-1 USER**

Attribute Usage Matrix:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attributes**  **Queries** | **A1**  **Aadhar\_no** | **A2**  **First\_name** | **A3**  **Last\_name** | **A4**  **Email\_id** | **A5**  **phone\_no** |
| **Q1** | 0 | 0 | 0 | 0 | 0 |
| **Q2** | 0 | 0 | 0 | 0 | 0 |
| **Q3** | 0 | 0 | 0 | 0 | 0 |
| **Q4** | 0 | 0 | 0 | 0 | 0 |
| **Q5** | 1 | 0 | 0 | 0 | 0 |
| **Q6** | 1 | 0 | 0 | 0 | 0 |
| **Q7** | 1 | 0 | 0 | 0 | 0 |

Attribute Affinity Matrix:

|  |  |
| --- | --- |
|  | **A1** |
| **A1** | 100 |

We have only one attribute which takes value in Attribute affinity matrix, Hence there is no need of cluster affinity matrix.

**Relation 2: Flight**

Attribute Usage Matrix:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes**  **Queries** | **A1**  **Flight\_id** | **A2**  **Departure\_city** | **A3**  **Arrival\_city** | **A4**  **fare** | **A5**  **Travel\_class** | **A6**  **Aadhar\_no** | **A7**  **Boarding\_date** | **A8**  **Boarding\_time** |
| **Q1** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **Q2** | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| **Q3** | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| **Q4** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **Q5** | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| **Q6** | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| **Q7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Attribute Affinity Matrix:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** | **A8** |
| **A1** | 50 | 50 | 50 | 0 | 0 | 0 | 0 | 0 |
| **A2** | 50 | 90 | 90 | 0 | 0 | 0 | 40 | 0 |
| **A3** | 50 | 90 | 180 | 0 | 0 | 70 | 0 | 0 |
| **A4** | 0 | 0 | 0 | 40 | 0 | 40 | 40 | 0 |
| **A5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A6** | 0 | 0 | 70 | 0 | 0 | 0 | 0 | 0 |
| **A7** | 0 | 40 | 40 | 40 | 0 | 40 | 80 | 0 |
| **A8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ordering the position of 3rd Column A3 of AA** | | | | | | |
| **Sr. No** | | **Contribution calculations** | | | | |
| 1 | | Cont(031)=2[Bond(03)+Bond(31)-Bond(01)] | | | | |
| Bond(03) = 0 | | Bond(31) = 50\*50+90\*50+180\*50+0\*0+0\*0+70\*0+40\*0+0\*0 =16,000 | | Bond(01) = 0 |
| **Cont(031) =** 2**\***(0+16,000-0) = **32,000** | | | | |
|  | | | | | | |
| 2 | | Cont(132)=2[Bond(13)+Bond(32)-Bond(12)] | | | | |
| Bond(13) = 50\*50+50\*90+50\*180+0\*0+0\*0+0\*70+0\*40+0\*0 =16,000 | | Bond(32) = 50\*50+90\*90+90\*180+0\*0+0\*0+0\*70+40\*40+0\*0 = 28,400 | Bond(12) = 50\*50+50\*90+ 50\*90+0\*0+0\*0+0\*40+0\*0 = 11,500 | |
| **Cont(132) =** 2\*(16,000+28,400-11,500) = **65,800** | | | | |
|  | | | | | | |
| 3 | Cont(234)=2[Bond(23)+Bond(34)-Bond(24)] | | | | | |
| Bond(23) = 50\*50+90\*90+90\*180+0\*0+0\*0+0\*70+40\*40+0\*0 = 28,400 | | | Bond(34) = 50\*0+90\*0+180\*0+0\*40+0\*0+0\*40+40\*40+0\*0=1,600 | Bond(24) = 40\*0+90\*0+90\*0+0\*0+0\*40+0\*0+40\*40+0\*0 = 1,600 | |
| **Cont(234) =** 2\*(28,400+1,600-1,600) = **28,400** | | | | | |
| **Maximum CONT =** | | | **65,800** | | | |
| **Since CONT(132) value is maximum so place A3 in between A1 and A2.** | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordering the position of 4th Column of AA** | | | | |
| **Sr. No.** | **Contribution Calculations** | | | |
| 1 | Cont(041)=2[Bond(04)+Bond(41)-Bond(01)] | | | |
| Bond(04) = 0 | | Bond(41) = 0 | Bond(01) = 0 |
| **Cont(041) =** 2**\***(0+0-0) = 0 | | | |
|  | | | | |
| 2 | Cont(142)=2[Bond(14)+Bond(42)-Bond(12)] | | | |
| Bond(14) = 0 | | Bond(42) = 1,600 | Bond(12) = 11,500 |
| **Cont(142) =** 2**\***(0+1,600-11,500) = **-19,800** | | | |
|  | | | | |
| 3 | Cont(243)=2[Bond(24)+Bond(43)-Bond(23)] | | | |
| Bond(24) = 1600 | | Bond(43) = 1600 | Bond(23) = 28400 |
| **Cont(243) =** 2**\***(1,600+1,600-28,400) = **-50,400** | | | |
|  | | | | |
| 4 | Cont(345)=2[Bond(34)+Bond(45)-Bond(35)] | | | |
| Bond(34) = 1600 | | Bond(45) = 0 | Bond(35) = 0 |
| **Cont(345) =** 2**\***(1600+0-0) = **3200** | | | |
| **Maximum CONT =** | | **3200** | | |
| **Since CONT(345) value is max so place A4 in between A3 and A5.** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordering the position of 6th Column(dob) of AA** | | | | |
| **Sr. No.** | **Contribution Calculations** | | | |
| 1 | Cont(061)=2[Bond(06)+Bond(61)-Bond(01)] | | | |
| Bond(06) = 0 | | Bond(61) = 3500 | Bond(01) = 0 |
| **Cont(061) = 7000** | | | |
|  | | | | |
| 2 | Cont(162)=2[Bond(16)+Bond(62)-Bond(12)] | | | |
| Bond(16) = 3500 | | Bond(62) = 7900 | Bond(12) = 11500 |
| **Cont(162) =** -200 | | | |
|  | | | | |
| 3 | Cont(263)=2[Bond(26)+Bond(63)-Bond(23)] | | | |
| Bond(26) = 7900 | | Bond(63) = 14200 | Bond(23) = 28400 |
| **Cont(263) =** -12600 | | | |
|  | | | | |
| 4 | Cont(364)=2[Bond(36)+Bond(64)-Bond(34)] | | | |
| Bond(36) = 14200 | | Bond(64) = 3200 | Bond(34) =1600 |
| **Cont(364) =** -31600 | | | |
|  | | | | |
| 5 | Cont(465)=2[Bond(46)+Bond(65)-Bond(45)] | | | |
| Bond(46) = 3200 | | Bond(65) = 0 | Bond(45) = 0 |
| **Cont(465) =** 6400 | | | |
|  | | | | |
| 6 | Cont(567)=2[Bond(56)+Bond(67)-Bond(57)] | | | |
| Bond(56) = 0 | | Bond(67) = 4800 | Bond(57) = 0 |
| **Cont(567) = 9600** | | | |
| **Maximum CONT =** | | **9600** | | |
| **Since maximum CONT(567) hence place A6 between A5 and A7.** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordering the position of 7th Column(h\_no) of AA** | | | | |
| **Sr. No.** | **Contribution Calculations** | | | |
| 1 | Cont(071)=2[Bond(07)+Bond(71)-Bond(01)] | | | |
| Bond(07) = 0 | | Bond(71) = 2000 | Bond(01) = 0 |
| **Cont(071) = 2000** | | | |
|  | | | | |
| 2 | Cont(172)=2[Bond(17)+Bond(72)-Bond(12)] | | | |
| Bond(17) = 2000 | | Bond(72) = 6800 | Bond(12) = 11500 |
| **Cont(172) =** -5400 | | | |
|  | | | | |
| 3 | Cont(273)=2[Bond(27)+Bond(73)-Bond(23)] | | | |
| Bond(27) = 6800 | | Bond(73) = 6800 | Bond(23) =28400 |
| **Cont(273) =** -29600 | | | |
|  | | | | |
| 4 | Cont(374)=2[Bond(37)+Bond(74)-Bond(34)] | | | |
| Bond(37) = 6800 | | Bond(74) = 4800 | Bond(34) = 1600 |
| **Cont(374) =** 20000 | | | |
|  | | | | |
| 5 | Cont(475)=2[Bond(47)+Bond(75)-Bond(45)] | | | |
| Bond(47) = 4800 | | Bond(75) = 0 | Bond(45) = 0 |
| **Cont(475) =** 9600 | | | |
|  | | | | |
| 6 | Cont(576)=2[Bond(57)+Bond(76)-Bond(56)] | | | |
| Bond(57) = 0 | | Bond(76) = 4800 | Bond(56) = 0 |
| **Cont(576) = 9600** | | | |
|  |
| 7 | Cont(678)=2[Bond(67)+Bond(78)-Bond(68)] | | | |
| Bond(67) = 4800 | | Bond(78) = 0 | Bond(68) = 0 |
| **Cont(678) = 9600** | | | |
| **Maximum CONT =** | | **20000** | | |
| **Since CONT(374) has max value hence place A7 in between A3 and A4.** | | | | |

So, Cluster Affinity Matrix is:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A1** | **A3** | **A2** | **A7** | **A4** | **A5** | **A6** | **A8** |
| **A1** | 50 | 50 | 50 | 0 | 0 | 0 | 0 | 0 |
| **A3** | 50 | 180 | 90 | 0 | 0 | 0 | 70 | 0 |
| **A2** | 50 | 90 | 90 | 40 | 0 | 0 | 0 | 0 |
| **A4** | 0 | 0 | 0 | 40 | 40 | 0 | 40 | 0 |
| **A7** | 0 | 40 | 40 | 80 | 40 | 0 | 40 | 0 |
| **A5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A6** | 0 | 70 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Partitioning of relation:

TA - set of attributes in f1

BA - set of attributes in f2

TQ - number of applications access only TA

BQ - number of application access only BA

OQ - number of application access both TA and BA

CTQ - total number of access to attribute by application that access only TA

CBQ - total number of access to attribute by application that access only BA

COQ - total number of access to attribute by application that access both TA and BA

Max (Z) = (CTQ\*CBQ-(COQ 2))

TA= {A1 }

BA= {A3, A2, A7, A4,A5, A6, A8 }

TQ= {}

BQ= {}

OQ= {Q5,Q6, Q7}

CTQ=50

CBQ=0

COQ=100

Z=50\*0-(100)2= -10000

--------------------------

TA= {A1, A3}

BA= {A2, A7, A4, A5, A6, A8 }

TQ= {}

BQ= {}

OQ= {Q5,Q6, Q7}

CTQ=0

CBQ=0

COQ=40

Z=0\*0-(40)2= -1600

--------------------------

TA= {A1, A3, A2}

BA= {A7, A4, A5, A6, A8 }

TQ= { Q5,Q6, Q7}

BQ= {}

OQ= {}

CTQ=40

CBQ=0

COQ=0

Z=40\*0-(0)2=0

--------------------------

TA= {A1, A3, A2, A7}

BA= { A4, A5, A6,A8 }

TQ= { Q5,Q6, Q7}

BQ= {}

OQ= {}

CTQ=40

CBQ=0

COQ=0

Z=40\*0-(0)2=0

--------------------------

TA= {A1, A3, A2, A7, A4 }

BA= {A5, A6 ,A8 }

TQ= { Q5,Q6, Q7}

BQ= {}

OQ= {}

CTQ=40

CBQ=0

COQ=0

Z=40\*0-(0)2=0

--------------------------

TA= {A1, A3, A2, A7, A4, A5}

BA= {A6, A8}

TQ= { Q5,Q6, Q7}

BQ= {}

OQ= {}

CTQ=40

CBQ=0

COQ=0

Z=40\*0-(0)2=0

Since Maximum Z valueis **0** there is no need to vertically fragment this relation.

**RELATION 3: CAB**

Attribute Usage Matrix:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attributes**  **Queries** | **A1**  **Cab\_no** | **A2**  **Driver\_name** | **A3**  **Driver\_phone\_no** | **A4**  **Aadhar\_no** | **A5**  **Pickup\_location** | **A6**  **Pickup\_time** |
| **Q1** | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q2** | 1 | 1 | 1 | 1 | 1 | 1 |
| **Q3** | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q4** | 1 | 0 | 0 | 0 | 1 | 0 |
| **Q5** | 1 | 0 | 0 | 0 | 1 | 0 |
| **Q6** | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q7** | 0 | 0 | 0 | 0 | 0 | 0 |

Attribute Affinity Matrix:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** |
| **A1** | 80 | 40 | 40 | 40 | 80 | 40 |
| **A2** | 40 | 40 | 40 | 40 | 40 | 40 |
| **A3** | 40 | 40 | 40 | 40 | 40 | 40 |
| **A4** | 40 | 40 | 40 | 40 | 40 | 40 |
| **A5** | 80 | 40 | 40 | 40 | 80 | 40 |
| **A6** | 40 | 40 | 40 | 40 | 40 | 40 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ordering the position of 3rd Column A3 of AA** | | | | | | |
| **Sr. No** | | **Contribution calculations** | | | | |
| 1 | | Cont(031)=2[Bond(03)+Bond(31)-Bond(01)] | | | | |
| Bond(03) = 0 | | Bond(31) = 12800 | | Bond(01) = 0 |
| **Cont(031) =** 25600 | | | | |
|  | | | | | | |
| 2 | | Cont(132)=2[Bond(13)+Bond(32)-Bond(12)] | | | | |
| Bond(13) = 12800 | | Bond(32) = 9600 | Bond(12) = 12800 | |
| **Cont(132) =** 9600 | | | | |
|  | | | | | | |
| 3 | Cont(234)=2[Bond(23)+Bond(34)-Bond(24)] | | | | | |
| Bond(23) = 9600 | | | Bond(34) = 9600 | Bond(24) = 9600 | |
| **Cont(234) =** 9600 | | | | | |
| **Maximum CONT =** | | | **25600** | | | |
| **Since CONT(031) has max value. Hence place A3 at the beginning.** | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ordering the position of 4th Column A4 of AA** | | | | | | |
| **Sr. No** | | **Contribution calculations** | | | | |
| 1 | | Cont(043)=2[Bond(04)+Bond(43)-Bond(03)] | | | | |
| Bond(04) = 0 | | Bond(43) = 9600 | | Bond(03) = 0 |
| **Cont(043) =** 19200 | | | | |
|  | | | | | | |
| 2 | | Cont(341)=2[Bond(34)+Bond(41)-Bond(31)] | | | | |
| Bond(34) = 9600 | | Bond(41) = 12800 | Bond(31) = 12800 | |
| **Cont(341) =** 19200 | | | | |
|  | | | | | | |
| 3 | Cont(142)=2[Bond(14)+Bond(42)-Bond(12)] | | | | | |
| Bond(14) = 12800 | | | Bond(42) = 9600 | Bond(12) = 12800 | |
| **Cont(243) =** 19200 | | | | | |
| **Maximum CONT =** | | | **19200** | | | |
| **Since all CONT value is 19200. Hence no need to change, place A4 as it is.** | | | | | | |

.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordering the position of 5th Column(mobile\_no) of AA** | | | | |
| **Sr. No.** | **Contribution Calculations** | | | |
| 1 | Cont(053)=2[Bond(05)+Bond(53)-Bond(03)] | | | |
| Bond(05) = 0 | | Bond(53) = 9600 | Bond(03) = 0 |
| **Cont(053) = 19200** | | | |
|  | | | | |
| 2 | Cont(351)=2[Bond(35)+Bond(51)-Bond(31)] | | | |
| Bond(35) = 9600 | | Bond(51) = 17600 | Bond(31) = 9600 |
| **Cont(351) =** 35200 | | | |
|  | | | | |
| 3 | Cont(152)=2[Bond(15)+Bond(52)-Bond(12)] | | | |
| Bond(15) = 17600 | | Bond(52) = 12800 | Bond(13) = 9600 |
| **Cont(152) =41600** | | | |
|  | | | | |
| 4 | Cont(254)=2[Bond(25)+Bond(54)-Bond(24)] | | | |
| Bond(25) = 12800 | | Bond(54) = 12800 | Bond(34) = 9600 |
| **Cont(254) =** 19200 | | | |
|  | | | | |
| 5 | Cont(456)=2[Bond(45)+Bond(56)-Bond(46)] | | | |
| Bond(45) = 12800 | | Bond(56) = 12800 | Bond(46) = 9600 |
| **Cont(456) = 32000** | | | |
| **Maximum CONT =** | | **41600** | | |
| **Since max CONT(152) has value. so place A5 in between A1 and A2.** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordering the position of 6th Column(dob) of AA** | | | | |
| **Sr. No.** | **Contribution Calculations** | | | |
| 1 | Cont(063)=2[Bond(06)+Bond(63)-Bond(03)] | | | |
| Bond(06) = 0 | | Bond(63) = 9600 | Bond(03) = 0 |
| **Cont(061) = 19200** | | | |
|  | | | | |
| 2 | Cont(361)=2[Bond(36)+Bond(61)-Bond(31)] | | | |
| Bond(36) = 9600 | | Bond(62) = 12800 | Bond(31) = 12800 |
| **Cont(361) =** 19200 | | | |
|  | | | | |
| 3 | Cont(162)=2[Bond(16)+Bond(62)-Bond(12)] | | | |
| Bond(16) = 12800 | | Bond(62) = 9600 | Bond(12) = 12800 |
| **Cont(162) =** 19200 | | | |
|  | | | | |
| 4 | Cont(264)=2[Bond(26)+Bond(64)-Bond(24)] | | | |
| Bond(26) = 9600 | | Bond(64) = 9600 | Bond(24) =9600 |
| **Cont(264) =** 19200 | | | |
|  | | | | |
| 5 | Cont(465)=2[Bond(46)+Bond(65)-Bond(45)] | | | |
| Bond(46) = 9600 | | Bond(65) = 12800 | Bond(45) = 12800 |
| **Cont(465) =** 19200 | | | |
| **Maximum CONT =** | | **19200** | | |
| **Since maximum CONT = 19200 same for all hence place A6 at same place.** | | | | |

Cluster Affinity Matrix is:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A3** | **A1** | **A5** | **A2** | **A4** | **A6** |
| **A3** | 40 | 40 | 40 | 40 | 40 | 40 |
| **A1** | 40 | 80 | 80 | 40 | 40 | 40 |
| **A5** | 40 | 80 | 80 | 40 | 40 | 40 |
| **A2** | 40 | 40 | 40 | 40 | 40 | 40 |
| **A4** | 40 | 40 | 40 | 40 | 40 | 40 |
| **A6** | 40 | 40 | 40 | 40 | 40 | 40 |

Partitioning of relation:

TA - set of attributes in f1

BA - set of attributes in f2

TQ - number of applications access only TA

BQ - number of application access only BA

OQ - number of application access both TA and BA

CTQ - total number of access to attribute by application that access only TA

CBQ - total number of access to attribute by application that access only BA

COQ - total number of access to attribute by application that access both TA and BA

Max (Z) = (CTQ \*CBQ - (COQ) 2)

TA= {A3}

BA= {A1,A5, A2, A4,A6}

TQ= {}

BQ= {Q1,Q2,Q3,Q5,Q6,Q7}

OQ= {}

CTQ=0

CBQ=110

COQ=0

Z=0\*110-(0\*0)=0

----------------------------

TA= {A3, A1}

BA= {A5, A2, A4,A6}

TQ= { Q1,Q2,Q3,Q5,Q6,Q7}

BQ= {}

OQ= {Q3}

CTQ=70

CBQ=0

COQ=40

Z=70\*0-(40)2=-1600

--------------------------

TA= {A3, A1, A5}

BA= {A2, A4,A6}

TQ= { Q1,Q2,Q3,Q5,Q6,Q7}

BQ= {}

OQ= {}

CTQ=40

CBQ=0

COQ=0

Z=0

--------------------------

TA= {A3, A1, A5, A2}

BA= { A4, A6}

TQ= {Q1,Q2,Q3,Q5,Q6,Q7}

BQ= {}

OQ= {}

CTQ=40

CBQ=0

COQ=80

Z=0

--------------------------

TA= {A3, A1, A5, A2, A4}

BA= {A6}

TQ= { Q1,Q2,Q3,Q5,Q6,Q7}

BQ= {}

OQ= {}

CTQ=20

CBQ=30

COQ=80

Z=-1200

--------------------------

Since Maximum Z value is zero. Hence no need to fragment this relation.

**Relation 4: HOTEL**

Attribute Usage Matrix:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes**  **Queries** | **A1**  **Hotel\_id** | **A2**  **Hotel\_name** | **A3**  **fare** | **A4**  **Location** | **A5**  **Aadhar\_no** | **A6**  **Check\_in\_date** | **A7**  **Check\_out\_date** |
| **Q1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q3** | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| **Q4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q5** | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| **Q6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Attribute Affinity Matrix:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| **A1** | 70 | 20 | 0 | 70 | 0 | 0 | 0 |
| **A2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A4** | 70 | 20 | 0 | 70 | 0 | 0 | 0 |
| **A5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordering the position of 3th Column of AA** | | | | |
| **Sr. No.** | **Contribution Calculations** | | | |
| 1 | Cont(031)=2[Bond(03)+Bond(31)-Bond(01)] | | | |
| Bond(03) = 0 | | Bond(31) = 0 | Bond(01) = 0 |
| **Cont(031) =0** | | | |
|  | | | | |
| 2 | Cont(132)=2[Bond(13)+Bond(32)-Bond(12)] | | | |
| Bond(13) = 0 | | Bond(32) = 0 | Bond(12) = 2800 |
| **Cont(132) =** -5600 | | | |
| **Maximum CONT =** | | **0** | | |
| **Since maximum CONT value is 0 so place A3 at same place.** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ordering the position of 4th Column of AA** | | | | |
| **Sr. No.** | **Contribution Calculations** | | | |
| 1 | Cont(041)=2[Bond(04)+Bond(41)-Bond(01)] | | | |
| Bond(04) = 0 | | Bond(41) = 9600 | Bond(01) = 0 |
| **Cont(041) =19200** | | | |
|  | | | | |
| 2 | Cont(142)=2[Bond(14)+Bond(42)-Bond(12)] | | | |
| Bond(14) = 9600 | | Bond(42) = 2800 | Bond(12) = 2800 |
| **Cont(142) =** 19200 | | | |
|  | | | | |
| 3 | Cont(243)=2[Bond(24)+Bond(43)-Bond(23)] | | | |
| Bond(24) = 2800 | | Bond(43) = 0 | Bond(23) = 0 |
| **Cont(243) =** 5600 | | | |
|  | | | | |
| 4 | Cont(345)=2[Bond(34)+Bond(45)-Bond(35)] | | | |
| Bond(34) = 0 | | Bond(45) = 0 | Bond(35) = 0 |
| **Cont(345) = 0** | | | |
| **Maximum CONT =** | | **19200** | | |
| **Since maximum CONT value is 19200 so place A4 between A1 and A2.** | | | | |

Since A5, A6, A7 have all values as zero in AAA, as there Bond and Contribution will also be zero. So need to calculate Bond and Maximum Contribution and it won’t affects the Cluster Affinity matrix.

Cluster Affinity Matrix:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A1** | **A4** | **A2** | **A3** | **A5** | **A6** | **A7** |
| **A1** | 70 | 70 | 20 | 0 | 0 | 0 | 0 |
| **A4** | 70 | 70 | 20 | 0 | 0 | 0 | 0 |
| **A2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

TA= {A1}

BA= {A4, A2, A3, A5,A6,A7}

TQ= {}

BQ= {}

OQ= {Q1,Q2,Q3,Q4}

CTQ=0

CBQ=0

COQ=160

Z=0\*0-(1602) =-25600

----------------------------

TA= {A1, A4}

BA= {A2, A3, A5,A6,A7}

TQ= {Q1,Q2,Q3}

BQ= {}

OQ= {Q4}

CTQ=140

CBQ=0

COQ=20

Z=140\*0-(20)2= -400

--------------------------

TA= {A1, A4, A2}

BA= {A3, A5,A6,A7}

TQ= {Q1,Q2,Q3}

BQ= {}

OQ= {Q4}

CTQ=140

CBQ=0

COQ=20

Z=140\*0-(20)2= -400

--------------------------

TA= {A1, A4, A2, A3}

BA= {A5,A6,A7}

TQ= {Q1,Q2,Q3}

BQ= {}

OQ= {Q4}

CTQ=140

CBQ=0

COQ=20

Z=140\*0-(20)2= -400

-----------------------------------

TA= {A1, A4, A2, A3,A5}

BA= {A6,A7}

TQ= {Q1,Q2,Q3}

BQ= {}

OQ= {Q4}

CTQ=140

CBQ=0

COQ=20

Z=140\*0-(20)2= -400

--------------------------

TA= {A1, A4, A2, A3, A5,A6}

BA= {A7}

TQ= {Q1,Q2,Q3,Q4}

BQ= {}

OQ= {}

CTQ=160

CBQ=0

COQ=0

Z=160\*0-(0)2= 0

-----------------------------------

Since maximum Z value is 0, there is no need to vertically fragment this relation.

**Relation 5: HOMESTAY**

Attribute Usage Matrix:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes**  **Queries** | **A1**  **Homestay\_id** | **A2**  **Homestay\_name** | **A3**  **fare** | **A4**  **Location** | **A5**  **Aadhar\_no** | **A6**  **Check\_in\_date** | **A7**  **Check\_out\_date** | **No\_of\_rooms** |
| **Q1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q4** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q7** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Attribute Affinity Matrix:

|  |  |  |
| --- | --- | --- |
|  | **A1** | **A5** |
| **A1** | 20 | 0 |
| **A2** | 0 | 0 |
| **A3** | 0 | 0 |
| **A4** | 0 | 0 |
| **A5** | 0 | 40 |

We have only two attributes which takes value in Attribute affinity matrix, Hence there is no need of cluster affinity matrix.

.

**Relation 6: PAYMENT**

Attribute Usage Matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attributes**  **Queries** | **A1**  **Payment\_id** | **A2**  **Card\_no** | **A3**  **Card\_holder\_name** | **A4**  **Card\_password** |
| **Q1** | 0 | 0 | 0 | 0 |
| **Q2** | 0 | 0 | 0 | 0 |
| **Q3** | 0 | 0 | 0 | 0 |
| **Q4** | 0 | 0 | 0 | 0 |
| **Q5** | 0 | 0 | 0 | 0 |
| **Q6** | 1 | 0 | 0 | 0 |
| **Q7** | 1 | 0 | 0 | 0 |

Attribute Affinity Matrix:

|  |  |
| --- | --- |
|  | **A1** |
| **A1** | 80 |

We have only one attribute which takes value in Attribute affinity matrix, Hence there is no need of cluster affinity matrix.

**PHYSICAL DESIGN**

Now, we will consider storing the fragments on the disk. This along with other parameters discussed later in this section is needed to compute local and remote, query and update times.

**Assumptions:**

First, let us assume the size of all the attributes of all relations to obtain the size of single record (tuple) of every relation.

Fixed length records are considered and records are spanned. The delimiter for each field is the length of the field. Integer = 4B & Date = 3B.

USER ----Fragment F1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aadhar\_no  (10) | FirstName  (20) | LastName  (20) | email\_id  (20) | phone\_no  (10) |

FLIGHT ----Fragment F2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Flight\_id  (10) | Departure\_city  (20) | Arrival\_city  (20) | Fare  (10) | Travel\_class  (10) | Aadhar\_no  (10) | boarding \_date  (10) | boarding \_time  (10) |

CAB ----Fragment F3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cab\_no  (10) | driver\_name  (20) | driver\_phone\_no  (10) | Aadhar\_no  (10) | Pickup\_location  (20) | Pickup\_time  (20) |

HOTEL ----Fragment F4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hotel\_id  (10) | Hotel\_name  (10) | Fare  (10) | Location((10) | Aadhar\_no  (10) | Check\_in\_date4(10) | Check\_out\_date  (10) |

HOMESTAY ----Fragment F5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Homestay\_no (10) | Homestay\_name (20) | Fare  (10) | Location  (10) | Aadhar\_no  (10) | Check\_in\_date  (10) | Check\_out\_date  (10) | no\_of\_rooms  (10) |

PAYMENT ----Fragment F6

|  |  |  |  |
| --- | --- | --- | --- |
| payment\_id  (10) | card\_no  (10) | card\_holder\_name  (10) | card\_password  (10) |

Secondly, let us consider the number of records (tuples) in each relation and number of blocks required to store each relation. For this we need to have the block size. Assume block size is 1024B and it is assumed that records span multiple blocks.

There are totally 7 fragments, each fragments size is mentioned in the below table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fragment** | **No. of Records** | **Single Record Size(B)** | **Total Size(B)** | **No. Of Blocks** |
| **F1** | 3000 | 120 | 360000 | 352 |
| **F2** | 200 | 20 | 4000 | 4 |
| **F3** | 500 | 40 | 20000 | 20 |
| **F4** | 3000 | 70 | 210000 | 206 |
| **F5** | 2000 | 60 | 120000 | 118 |
| **F6** | 1500 | 50 | 75000 | 74 |

**DISK PARAMETERS**

Average Seek Time(S) = 10 ms

Average Latency Time (L) = 4 ms

Inter-Block gap-size (g) = 100 bytes

Block transfer rate (Btr) = 0.5 ms

Block pointer size = 6 bytes

List of Formulas

Access time = (S + Btr) \*N, Where N is number of blocks for a fragment

Transfer rate = Block size / Btr

= (1024 / 0.5)

= 2 Blocks

This is without considering Interleaving gap

With interleaving gap

Block Transfer with interleaving gap (Tr) = B / (B+G) G-Interleaving gap

=> 1024 / (1024+106) \* 2048

= 1856bytes

1856B ----------- 1ms

1024B ----------- 1024 / 1856 = 0.55ms

Block access time (Tr) = 0.55ms

Therefore,

To Retrieve fragments of N blocks (locally) = (S + L + Tr) \* N

To Write fragments of N blocks update-time (locally) = 2 \* (S + L + Tr) \* N

(2\* is included in the Update time, since the data block has to be fetched into memory from the disk, updated and then written back to the disk)

**REPLICATION AND ALLOCATION**

**Assumptions and List of Formulae:**

Propagation Delay (Tp) = Distance between sites / Speed of transmission media

Transmission Delay (Td) = Packet Size / Bandwidth

Where,

Speed of transmission media = 2.7 \* 104 m/s

Packet Size = 1024B

Bandwidth = 1024 KB/s

It is assumed that a packet is the smallest unit used to send/receive data. If the data is small then the packet is padded to make it equal to 1024B.

Remote Retrieval Time = Local Retrieval Time + Td + (2 \* Tp)

Remote Update Time = Local Update Time + (2 \* Tp)

For Remote Retrieval Time, Transmission delay (Td) is added once only because the size of the query is very small and hence negligible. The time is taken for the transmission of the data that is being fetched, the time taken for query to propagate to the remote site and the data to propagate back (hence 2 \* Tp).

For Remote Update Time, the Transmission Delay for the query and the acknowledgement (that the update was successful) is negligible. Thus, only the Propagation Delay is considered here.

From the above Formulas and Assumptions, and considering the distance, we get the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **From Site** | **To Site** | **Distance(km)** | **Propagation Delay (ms)** | **Transmission Delay (ms)** |
| S1 | S2 | 270 | 100 | 1 |
| S1 | S3 | 270 | 100 | 1 |
| S2 | S3 | 270 | 100 | 1 |

From the above Formulas and Assumptions, and considering the distance, we get the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fragments** | **Local Retrieval Time(ms)** | **Remote Retrieval Time(ms)** | **Local Update Time(ms)** | **Remote Update Time(ms)** |
| **F1** | 4845.15 | 5046.15 | 9690.3 | 9890.3 |
| **F2** | 174.6 | 375.6 | 349.2 | 549.2 |
| **F3** | 1571.4 | 1772.4 | 3142.8 | 3342.8 |
| **F4** | 3201 | 3402 | 6402 | 6602 |
| **F5** | 9952.2 | 10153.2 | 19904.4 | 20104.4 |
| **F6** | 931.2 | 1132.2 | 1862.4 | 2062.4 |

For Allocation, we are using Redundant All Beneficial Site Method.

|  |  |  |  |
| --- | --- | --- | --- |
| **Transactions (Queries)** | **Site** | **Frequency** | **Fragment Access** |
| **Q1** | S2 | 50 | F2:1st,0 wr  F3:1st, 0 wr |
| **Q2** | S2 | 50 | F2: 1st, 0 wr  F3: 1st , 0 wr |
| **Q3** | S2, S3 | 40,30 | F1:1st, 0 wr  F2: 1st,0 wr  F3: 2nd , 0 wr  F4: 2nd, 0 wr |
| **Q4** | S2 | 20 | F2: 1rd,0 wr  F3: 1rd,0 wr |
| **Q5** | S1,S2 ,S3 | 40, 30,50 | F1:1rd, 0 wr  F2:1rd, 0 wr  F4:2rd, 0 wr |
| **Q6** | S1,S2 | 70,30 | F2: 1st, 0 wr  F6: 1st, 0 wr |
| **Q7** | S1, S3 | 20,50 | F5: 2rd, 0 wr  F6: 1ST |

**Cost Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fragment** | **Site** | **Remote Update** | **Local Update** | **#writes\*frequency\*time** | **cost** |
| **F1** | S1 | - | - | 0 | 0 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | - | - | 0 | 0 |
| **F2** | S1 | - | - | 0 | 0 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | - | - | 0 | 0 |
| **F3** | S1 | - | - | 0 | 0 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | - | - | 0 | 0 |
| **F4** | S1 | - | - | 0 | 0 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | - | - | 0 | 0 |
| **F5** | S1 | - | - | 0 | 0 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | - | - | 0 | 0 |
| **F6** | S1 | - | - | 0 | 0 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | - | - | 0 | 0 |

**Benefit Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fragment** | **Site** | **Queries** | **#read\*frequency\*(remote-local time)** | **Benefit** | **Benefit-Cost** |
| **F1** | S1 | - | - | 0 | 0 |
|  | S2 | Q5, Q6, Q7 | 5\*40\*201 +2\*40\*201+1\*20\*201 | 30150 | 30150 |
|  | S3 | - | - | 0 | 0 |
| **F2** | S1 | - | - | 0 | 0 |
|  | S2 | Q1,Q2,Q3,Q5, Q6,Q7 | 2\*50\*201+2\*50\*201+  2\*40\*201+1\*20\*201 +3\*20\*201+2\*70\*201 | 80200 | 80200 |
|  | S3 | - | - | 0 | 0 |
| **F3** | S1 | - | - | 0 | 0 |
|  | S2 | Q2,Q4,Q5 | 1\*50\*201+2\*40\*201+  1\*20\*201 | 30150 | 30150 |
|  | S3 | - | - | 0 | 0 |
| **F4** | S1 | - | - | 0 | 0 |
|  | S2 | Q3,Q5 | 3\*50\*201+  2\*40\*201 | 40200 | 40200 |
|  | S3 | - | - | 0 | 0 |
| **F5** | S1 | Q4,Q7 | 3\*20\*201+2\*50\*201 | 40200 | 40200 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | - | - | 0 | 0 |
| **F6** | S1 | - | - | 0 | 0 |
|  | S2 | - | - | 0 | 0 |
|  | S3 | Q6, Q7 | 2\*40\*201+2\*30\*201 | 28140 | 28140 |

**Allocation:**

|  |  |
| --- | --- |
| **Site** | **Fragments** |
| **S1** | F5 |
| **S2** | F1, F2, F3, F4 |
| **S3** | F6 |

**INDEXING**

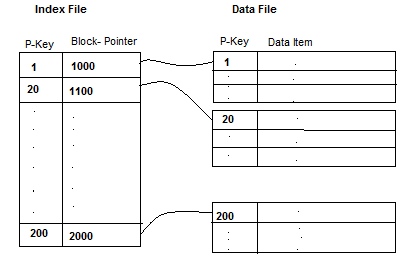
A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed.

Indexing is defined based on its indexing attributes. Indexing can be of the following types –

* Primary Index − Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.
* Secondary Index − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
* Clustering Index − Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

In our assumption, we have not considered indexing. Therefore, to access a fragment, all the blocks have to be accessed. For ex, to do a search in fragment F1, in the worst case, all of its 352 blocks has to be accessed. That is a major flaw of not using indexing. So, we will go for indexing now.

In the proposed system, most of the queries are accessing the tables using the primary key attribute. So, it is profitable to index the tables on the primary key. Therefore, we are going for Primary Indexing.



Assuming,

Block Pointer Size = 6B

Block Size = 1024B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fragment** | **No of blocks** | **size of index table entry** | **total size of index table** | **no of blocks** |
| **Fragment 1** | 352 | 10+6 | 5632 | 6 |
| **Fragment 2** | 4 | 10+6 | 64 | 1 |
| **Fragment 3** | 20 | 10+6 | 320 | 1 |
| **Fragment 4** | 206 | 10+6 | 3296 | 4 |
| **Fragment 5** | 118 | 10+6 | 1888 | 2 |
| **Fragment 6** | 74 | 10+6 | 1184 | 2 |

Hence from here we can see that we have reduced the no of block access required for fragment 1 from 352 to 7 using primary indexing. In the worst case we will have to access 7 blocks. 6 for the index blocks + 1 for the data block. Therefore it is an improvement in number of accessing blocks.

Same is the case with other Fragments.