NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

Department of Information Technology

Advanced Database Systems
Assignment 1

Online Shopping Management System.

Submitted by
Name: Shubham Ganvir (222IT011)
Krishna Kumar Munda(222IT016)

Class : M.Tech, I-semester

Contents

Assignment 1

Topic: Online Shopping Management System

Online shopping management system is developed to manage shopping system online. It is useful for sellers having outlets in different cities want to sell all over in country. Using this system many sellers can sell their products online reaching a wider range of audience. It will be easy for sellers to sell online and buyers to buy wide range of products.

We will break the problem into some smaller problems involved in online shopping management.

- Consider a user A which visits our online shopping system. He has to register on our portal. For this the 'USER' entity has details i.e. attributes like user_name, user_email_id, user_contact_no are stored in our system.

- Consider there is company B which wants to sell their products. All product related information will be stored on company B. So to maintain this company B keeps all products related information in entity "Product" like its id, type, year of manufacture, price, expiry date, country of origin. They can also keep the vendor related information like it's id, name, address, type of vendor.

- Consider there company C which provides payment services. It provides payment services for various customers. They provide interface and digital solutions for all kinds of payments. For this the 'PAYMENT' entity has details like reference id, date, time, mode of payment, amount, discount given etc are stored in their own database.

- Consider there is company D which provides logistic support services including delivery agent for delivery upto user's location.

So if a person wants to make online shopping system, he will collects the products related information from company B. When User A visits our system, he has to register for first time, after registering his information is stored in our system. Once the user A places order for a product, for payment, our system has to take help from C which

provides payment related services. For delivery of products to user, our system has to take help from company D for logistic related support which will include the delivery of products upto user's shipped address. We have to make User A, Products (company B),Payments (company C), Logistic service (company D) all these different role playing groups communicate with each other because of the data and control flow between multiple locations in online shopping management system. So that it feels like using a single standalone platform.

Data Sources :

1) User information: It has all user related information. The 'USER' entity has details i.e. attributes like user_name, user_email_id, user_address , user_mob_no are stored in our system (user information is stored in system once he visits or register on our system, hence it will acts as independent data source.)

2) Product information: It has different product related information which seller wants to sell. It's all information is stored in company A.

3) Payment information: All payment related information is maintained at specific location by company C.

4) Logistic Service information: All logistic related information is maintained at specific location by company B.

Actors :

1) Users
2) Sellers
3) Delivery agents

Queries :

● Queries:

1) Find all user information from a particular location x ?
2) Find all available vendors for a particular product name x ?
3) Find all users who has done the payment for all products?
4) Find users who checked the product name y and product price >300 ?

5) List the users who have given order on orderdate “2022-12-01”?
6) Find the vendor information for a product name z?
7) Find the orderdate which has maximum payment?
8) List the users who checks a product belonging to category y?

2. ENTITIES THAT ARE MAINTAINED IN DIFFERENT LOCATIONS

Entities	Location	Information
User, Orders	Site 1	This contains the information of user and orders which is given by users.
Product, Vendor, Category	Site 2	This contains product related information ,vendor supplying the product and products different category.
Payment	Site 3	This contains payment related information like payment date time and amount paid by users.
Courier	Site 4	This contains the courier related information like courier names for delivering a product.

Entities Sets :

User (user_name, user_id, user_email, user_contact_no, user_location)

Product (P_id, P_name, cat_id, Manufacture_year, vid ,quantity, price, user_id)

Categogy (cat_id, cat_name)

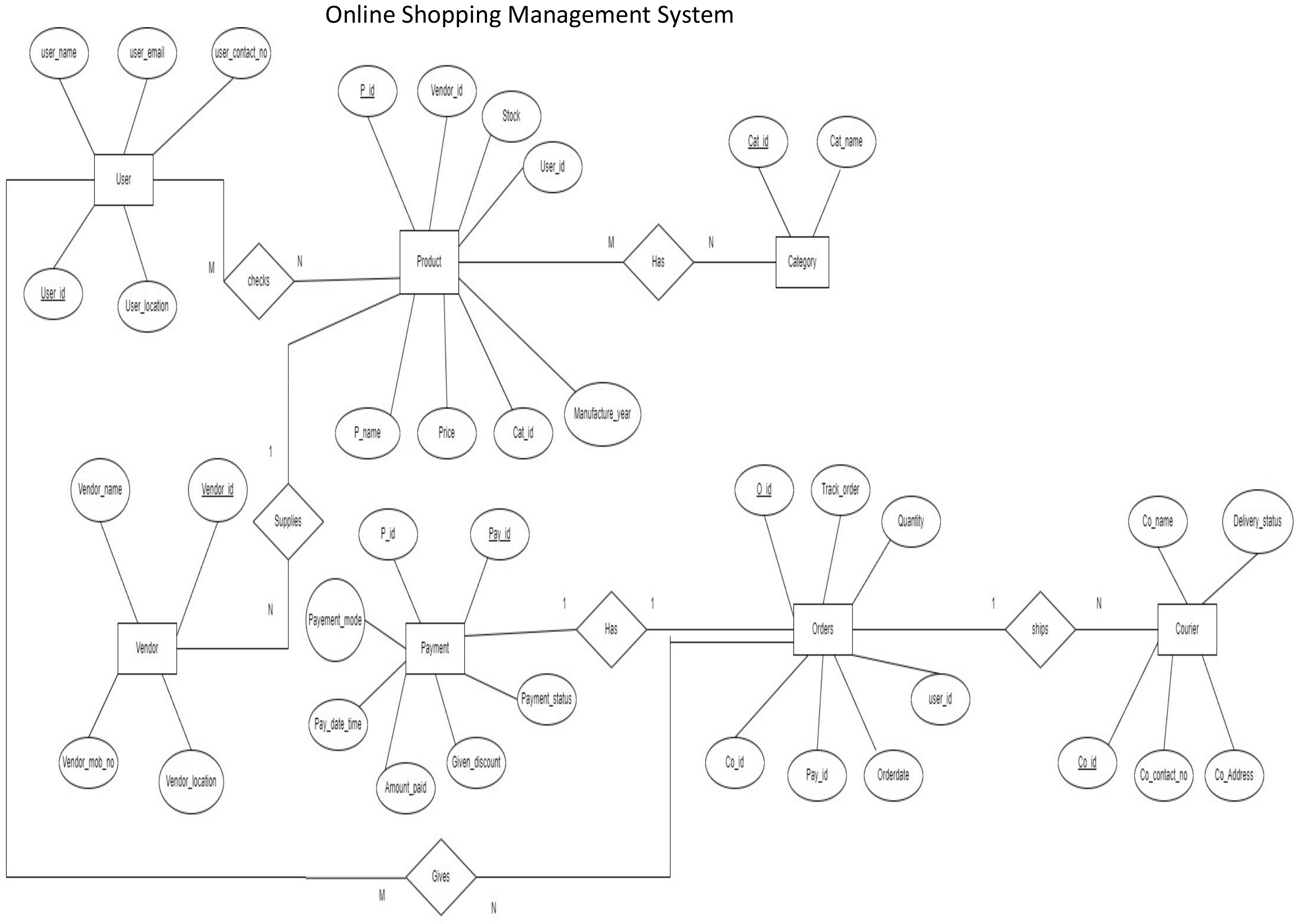
Vendor (Vendor_id, Vendor_name, Vendor_contact_no, Vendor_location, Delivery_status)

Payment (Pay_id, Aamount_paid, discount_given, pay_status, P_id, payment_mode)

Orders (o_id, co_id, Pay_id, Orderdate, quantity, Track_order, user_id)

Courier (Co_id, Co_name, Co_contact_no, Delivery_status, Co_location, Shipping_address)

Online Shopping Management System



4) Global Conceptual Schema:
   The Entity Relationship diagram is transformed into relational model. A feasible set of relational schema is as follows:

User:

User_id	User_name	User_email	User_contact_no	User_location

Product :

P_id	P_name	Cat_id	Vendor_id	Quntity	Price	Manufacture_year	Discount_given	User_id

Category:

Cat_id	Cat_name

Vendor:

Vendor_id	Vendor_name	Ven_contact_no	Vendor_location

Payment:

Pay_id	P_id	Pay_mode	Amount_paid	Discount_given	Pay_date_time

Orders:

O_id	Co_id	Orderdate	Trace_order	Pay_id	Quantity	User_id

Courrier

Co_id	Co_name	Co_phone_no	Shipping_Address	Delivery_status	Co_location

5. NORMALIZATION :

Some queries might have to travel with more than one table based on foreign key by some kind of joining. If we have 100’s of table then joining kind of operation will take a lot of time. So, it is undesirable. Alternative solution is keeping a universal or central table having all attributes together. It eases our information retrieval but there can be lots duplication or redundant values. Redundancy is repeated values in a same table, which leads to wastage of space, example, for 100 employees working for a same department; we have to repeat same department information for all 100 employees in a table. We have huge storage capacity at lower cost nowadays. So, what is the trouble here? Anomalies.

Anomalies are a kind of inconsistent information and overhead. Database won’t show any error and will simply accept the values. It results to imprecise output, example, from 100 employees in “sales” department, I changed department name to “production” by mistake for 10 employees. Now if I query for list of employees in sales department, it will show only 90 employees details, 10 are left out. Database won’t show any error for changing 10 employees detail but it simply accepts. Such kind of inconsistency is called anomaly. It can happen while insert, delete and updating information. 1. Insert anomaly: while inserting employee information we might enter wrong department details, which will lead to anomaly. 2. Delete anomaly: there are 100 employees in a department “sales”. If we delete all 100 employees in that department then we would lose department details too. There is no other way to extract department details of “sales”. 3. Update anomaly: while updating a table we might enter some other details by mistake, which will lead to anomaly.

These are the overheads, when you combine all tables into a centralized one. So, how can we eliminate them? Solution is dividing tables as smallest as possible. But, how small a table should be? To the level of anomalies wont occurs, ideally table having 2 attributes. But, achieving this will increase querying time. Therefore, how to determine the number of attributes to keep in table which won’t lead to large querying time and anomalies? Idea is dividing large table into small tables having less number of attributes in such a way that your design reduces anomalies and subsequently degree of redundancy that are present in the table. This systematic procedure is called normalization. Normalization is carried out having functional dependency and candidate keys in mind. It is a step-wise process that divides relations into several pieces until we eliminate redundancy and anomalies. There are many steps to achieve this normalization.

1) 1st normal form

2) 2 nd normal form

3) 3 rd normal form

4) Boyce-codd normal form

Each follows their own set of rules. Let's discuss with the help of tourism agency database management system's global conceptual schema. What is key in general? Given a value of an attribute, we must be able to uniquely identify all other attributes given in a table. That is, a complete row also called tuple or record.

Example, if I say 3 in the attribute A, then it must return BC values. -> is called functional dependency. The term "functional" implies group of attributes that determine another group of attributes. We can also say attribute level dependency instead of functional dependency. But, mathematically speaking this kind of representation (LHS->RHS) is called "functionally dependent". Therefore, applying mathematics (set theory) on table design helps us dealing attributes. There are 3 different kinds of FD's. 1. Trivial Functional Dependency (FD): What is got on RHS is already LHS

Ex: A determines itself.

A->A

A->AB

AB->A

Closure – how many attributes you are able to determine by one attribute. It is the base of entire normalization process. Using closure property we can determine candidate keys. Candidate keys are key or set of keys that identify all other attributes in a table. A table can have many candidate keys, but at any moment, only one candidate key can be as a primary key of a table. If there are n attributes then 2n−1 Candidate keys are possible except null. Sometimes, candidate key with non-key attribute uniquely determine a table. Such key is called super key. Minimal super key or candidate key which has less no of attributes is called primary key, which uniquely identifies a tuple. In FD's, LHS must be a key for every table. In BCNF, we have 0% redundancy in table. To achieve this, we go through series of normalization from 1st NF to BCNF. It is not mandatory to go from 1st NF to BCNF. But, it is a convention to follow this sequence. Every normal form should be lossless, and FD preserved 1 st Normal Form: A relation is said to be in first normal form then it should satisfy the following

✓ No multi-valued attribute.

No composite attribute ✓

Identify primary key Here, the relationship is converted to either relation or foreign key or merging relations. Foreign key: giving primary key of one table as a reference to another table.

User:

<u>User_id</u>	User_name	User_email	User_contact_no	User_location

Product :

P_id	P_name	Cat_id	Vendor_id	Quntity	Price	Manufacture_year	Discount_given	User_id

Category:

Cat_id	Cat_name

Vendor:

Vendor_id	Vendor_name	Ven_contact_no	Vendor_location

Payment:

Pay_id	P_id	Pay_mode	Amount_paid	Discount_given	Pay_date_time

Orders:

O_id	Co_id	Orderdate	Trace_order	Pay_id	Quantity	User_id

Courrier

Co_id	Co_name	Co_phone_no	Shipping_Address	Delivery_status	Co_location

Outcome of 1st normalization: ✓ Primary key has been identified in each table using closure property (minimal super key)

✓ Composite attributes has been resolved

✓ Multi-valued attributes has been resolved.

2nd Normal Form: A relation in 2nd normal form if it follows the below conditions.

▪ It is already in the second normal form.

▪ Every non-prime attribute is fully functionally dependent on the primary key.

Basically, we want to eliminate all the partial functional dependencies in our database. All our relations in the database are already in 2nd normal form because every relation has a single attribute primary key, due to which we can say that all non-prime attributes will be functionally dependent on the primary key. Hence, all our relations are already in 2nd Normal Form.

3rd Normal Form: For a relation to be in 3rd normal form it should satisfy the following conditions

▪ It should already be in 2nd Normal Form.

▪ The relation shouldn't contain any transitive dependencies: non-prime attributes transitively depending on the key.

3 rd Normal form should hold the condition, if X->Y then: Either X is a super key or Y is a prime attribute. By using this rule, we can eliminate all transitive functional dependencies. There are no transitive dependencies in our database so all are in 3rd Normal Form.

Global Schema:

● User

Attribute name	Size
User_id	Int(10)
User_name	Char(20)
User_location	Char(25)
User_email	Char(30)
User_Contact_no	Int(10)

● Product

Attribute name	Size
P_id	Int(10)
P_name	Char(25)
User_id	Int(10)
Price	Int(6)
Quantity	Int(10)
Manufacture_year	Date(3)
Vendor_id	Int(10)
Cat_id	Int(10)

● Category

Attribute name	Size
Cat_id	Int(10)
Cat_name	Char(20)

● Vendor

Attribute name	Size
Vendor_id	Int(10)
Vendor_name	Char(20)
Vendor_location	Char(25)
Vendor_Contact_no	Int(10)

● Orders

Attribute name	Size
O_id	Int(10)
Co_id	Int(10)
Orderdate	Date(3)
Quantity	Int(10)
Trace_order	Char (20)
User_id	Int(10)
Pay_id	Int(10)
O_id	Int(10)

● Payment

Attribute name	Size
Pay_id	Int(10)
P_id	Int(10)
Pay_mode	Char(15)
Amount_paid	Int()
Discount_given	Int()
Pay_date	Date(3)

● Courier

Attribute name	Size
Co_id	Int(10)
Co_name	Char(20)
Co_contact_no	Int(10)
Co_location	Char(25)
Shipping_address	Char(30)
Delivery_status	Char(10)

Fragmentation:

The main goal of DDBMS is to provide the data to the user involving less overhead and as quickly as possible. This is provided by fragmentation of data. Data Fragmentation provides distribution transparency of the data over the database. Dividing the whole table into smaller chunks and storing them in different Databases in the Distributed Database Management System is called data fragmentation. Fragmentation of data provides the following advantages:

• Storage will not be exhausted quickly.

• Provides parallel processing.

• Provides Load balancing.

• Improves query response time.

• Provides better local processing.

• Better availability of data. Decomposed fragments are placed into some other site to facilitate query and optimize other quality of services. These fragments permit a number of transactions concurrently. Taking a copy of a relation and maintaining it in another site is called replication. One can combine fragmentation and replication for better service provision. There are two kinds of fragmentation: horizontal and vertical. They must satisfy the following properties:

• Completeness: all rows or columns must be present in at least one site.

• Reconstruction: while reconstructing the relation, there should not be any inconsistency or loss of data.

• Dis-jointness: row or column must be present in at most one site, else will lead to inconsistent data. Fragmentation takes place in a relation based on the query and its frequency. The predicates used in the query servers are an important statistical input for fragments.

Following are the lists of queries depicting the transactions in the Online Shopping Management system.

Query 1)

Find the user_name, user_email for all users from a particular location ‘A’

Ans: Select user_name,user_email form user where user_location=’A’

Query 2)

List all available vendors for a product name ‘Laptop’

Ans: Select vendor_id, vendor_name from vendor,product where vendor.v_id=product.v_id and product.P_name=’Laptop’

Query 3)

Find all usernames and user contact numbers for users which has done a payments for prodcuts.

Ans: select User_name, User_contact_no from user,product,payment where user.user_id=product.user_id and product.P_id=Payment.P_id

Query 4)

List all username,useremail for users who checked the product name ‘Y’ having price >300

Ans: select user_name, user_email from Users, Product where user.user_id=product.user_id and product.P_name=’Y’ and product.price > 300

Query 5)

List all usernames ,user locations who have given order on orderdate=”Y’

Ans: Select User_name, User_location from user, Orders where user.user_id=Orders.user_id and

Orders.Orderdate=’Y’

Query 6)

Find the Courier_name , Courier_contact_no for all usernames who had ordered an order on orderdate ‘Z’

Ans: Select Courier_name ,Courier_contact_no from User,Orders,Courier where user.user_id=Orders.user_id and Orders.Co_id=Courier.Co_id and Orders.orderdate=’Z’

Query 7)

Find the Orderdate which has maximum payment.

Ans: Select Orderdate from Orders, Payment where Orders.Pay_id=Payment.Pay_id and

Payment.Amount_paid=(select max (amount_paid) from Payment)

Query 8)

Find all user_names , user contact numbers of users who checks a product having category_name ='Clothes'

Ans: Select User_name, User_contact_no from User,Product where user.user_id=Product.user_id and Product.Cat_id=Category.Cat_id and Category.Cat_name='Clothes'

Horizontal Fragmentation:

Horizontal fragmentation partitions the relation along its tuples of the relations. Every Fragment will have the same number of attributes. There are two ways doing it. Primary and Derived horizontal fragmentation. But, it is usually done using the predicate defined on the queries.

Examples: In the above mentioned queries, 1) is an example of horizontal fragmentation. There we are retrieving details of the user_name and user_email for a particular User_location 'A'

Vertical Fragmentation:

The vertical fragmentation of a relation R produces subschema's R1, R2, R3, , ,, , , Rn. Each of which contains subset of attributes, and only one fragment has candidate key. To satisfy reconstruction, we need to use a joining attribute common between the sub schemas. There are two methods to perform vertical fragmentation:

(i) grouping (bottom up): done by combining every two attributes at a time and takes a long time if number of attributes are over 100 to get desired fragments.

(ii) splitting (top down) : given all attributes together is taken as a fragment and split them as many fragments as you want to get. This is much quicker than the first method. 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 access. 3. Attribute Affinity Matrix specifies the affinity measure of each pair. The sites maintained in the above problem description are as below:

Users , Orders	Site 1
Product, Vendor, Category	Site 2
Payment	Site 3
Courier	Site 4

	S1	S2	S3	S4	
Q1	5	0	0	0	
Q2	0	15	0	0	
Q3	10	15	0	0	
Q4	5	10	0	0	
Q5	15	0	0	0	
Q6	10	0	0	15	
Q7	15	0	20	0	
Q8	5	8	0	0	

Relation-1: user

Attribute Usage Matrix:

User_name	User_id	User_email	User_contact_no	User_location
1	0	1	0	1
0	0	0	0	0
1	1	1	0	0
1	1	1	0	0
1	1	0	0	1
0	1	0	0	0
0	0	0	0	0
1	1	0	1	0

For User table:

```
attribute affinity matrix is
[[A1 A2  A3 A4 A5]
 [73 68 45 13 20]
 [68 93 40 13  15]
 [45 40 45 0  5]
 [13 13  0 13  0]
 [20 15  5 0  20]]
```

```
Best location for attribute = A3
cont( A 0 , A 3 , A 1 ) = 2 * ( 0  +  8130.0  -  0 ) =  16260.0
cont( A 1 , A 3 , A 2 ) = 2 * ( 8130.0  +  8655.0  -  13557.0 ) =  6456.0
cont( A 2 , A 3 , A 4 ) = 2 * ( 8655.0  +  1105.0  -  2262.0 ) =  17310.0
```

Here the contribution is maximum value of 17310
Hence the order is [A2 A3 A4] is chosen.

Ordering the position of 4th column A4 of AA:

```
cont( A 0 , A 4 , A 1 ) = 2 * ( 0  +  2002.0  -  0 ) =  4004.0
cont( A 1 , A 4 , A 2 ) = 2 * ( 2002.0  +  2262.0  -  13557.0 ) =  -18586.0
cont( A 2 , A 4 , A 3 ) = 2 * ( 2262.0  +  1105.0  -  8655.0 ) =  -10576.0
cont( A 3 , A 4 , A 5 ) = 2 * ( 1105.0  +  455.0  -  1825.0 ) =  2210.0
```

Here the contribution is maximum value of 4004.0

Hence the order is [A0,A4,A1] is chosen.

Ordering the position of 5th column A5 of AA:

```
cont( A 0 , A 5 , A 1 ) = 2 * ( 0  +  455.0  -  0 ) =  910.0
cont( A 1 , A 5 , A 2 ) = 2 * ( 455.0  +  3105.0  -  2002.0 ) =  3116.0
cont( A 2 , A 5 , A 3 ) = 2 * ( 3105.0  +  3255.0  -  13557.0 ) =  -14394.0
cont( A 3 , A 5 , A 4 ) = 2 * ( 3255.0  +  1825.0  -  8655.0 ) =  -7150.0
cont( A 4 , A 5 , A 6 ) = 2 * ( 1825.0  +  0  -  0 ) =  3650.0
```

Here the contribution is maximum value of 3650

Hence the order is [A4 A5 A6] is chosen.
Hence overall order is [A4 A1 A2 A3 A5]
CA =

```
[[ A4.  A1.  A2.  A3.  A5.]
 [13. 13. 13.  0.  0.]
 [13. 73. 68. 45. 20.]
 [13. 68. 93. 40. 15.]
 [ 0. 45. 40. 45.  5.]
 [ 0. 20. 15.  5. 20.]]
```

Partitioning:
TA - set of attributes in fragment f1
TB - set of attributes in fragment f2
TQ - Number of applications accesses only TA
BQ - Number of applications accesses only TB
OQ - Number of applications accesses both TA and TB
CTQ - Total number of access to attributes by applications that access only TA
CBQ - Total number of access to attributes by applications that access only TB
COQ - Total number of access to attributes by applications that access both TA and TB

Z= (CTQ*CBQ)-(COQ*COQ)
Select maximum Z value

Fragments = [4] [1 2 3 5]
TA = [A4]
TB = [A1,A2,A3,A5]
TQ = [ ]
BQ = [Q1,Q3,Q4, Q5, Q6]
OQ = [Q2, Q7, Q8]
z = [-4624.0]

Fragments = [A4 A1] [A2 A3 A5]
TA = [A4, A1]
TB = [A2, A3, A5]
TQ = []
BQ = [6]
OQ = [Q1, Q2, Q3, Q4, Q5, Q7, Q8]
z = [0]

```
Fragments = [A4, A1, A2] [A3, A5]
TA = [A4, A1, A2]
TB = [A3, A5]
TQ = [Q6, Q8]
BQ = []
OQ = [Q1, Q2, Q3, Q4, Q5, Q7]
z = [0]

Fragments = [A4 A1 A2 A3] [A5]
TA = [A4 A1 A2 A3]
TB = [A5]
TQ = [Q3, Q4, Q6, Q8]
BQ = []
OQ = [Q1, Q2, Q5, Q7]
z = [-5625.0]
```

Maximum Z value is 0. So fragmentation is application dependent.so we don’t want to do any fragmentation here.

For relation product:

```
Query Access Matrix =
 [A1 A2 A3 A4 A5 A6 A7 A8]
 [0 0 0 0 0 0 0 0]
 [0 1 0 0 0 1 0 0]
 [1 0 1 0 0 0 0 0]
 [0 1 1 1 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 1 0 1 0 0 0]]

Attribute affinity matrix =
[[ A1.  A2.  A3.  A4.  A5.  A6.  A7.  A8.]
 [25.  0. 25.  0.  0.  0.  0.  0.]
 [ 0. 30. 15. 15.  0. 15.  0.  0.]
 [25. 15. 53. 15. 13.  0.  0.  0.]
 [ 0. 15. 15. 15.  0.  0.  0.  0.]
 [ 0.  0. 13.  0. 13.  0.  0.  0.]
 [ 0. 15.  0.  0.  0. 15.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]]

Best location for attribute = A3
cont( A 0 , A 3 , A 1 ) = 2 * ( 0  +  1950.0  -  0 ) =  3900.0
cont( A 1 , A 3 , A 2 ) = 2 * ( 1950.0  +  1470.0  -  375.0 ) =  6090.0
cont( A 2 , A 3 , A 4 ) = 2 * ( 1470.0  +  1245.0  -  900.0 ) =  2940.0
Here the contribution is maximum value of 6090
Hence the order is [A1 A3 A2] is chosen.

Best location for attribute = A4

cont( A 0 , A 4 , A 1 ) = 2 * ( 0  +  375.0  -  0 ) =  750.0
cont( A 1 , A 4 , A 2 ) = 2 * ( 375.0  +  1245.0  -  1950.0 ) =  -660.0
cont( A 2 , A 4 , A 3 ) = 2 * ( 1245.0  +  900.0  -  1470.0 ) =  1350.0
cont( A 3 , A 4 , A 5 ) = 2 * ( 900.0  +  195.0  -  195.0 ) =  1800.0
Here the contribution is maximum value of 1800
Hence the order is [A3 A4 A5] is chosen.

Best location for attribute = A5
cont( A 0 , A 5 , A 1 ) = 2 * ( 0  +  325.0  -  0 ) =  650.0
cont( A 1 , A 5 , A 2 ) = 2 * ( 325.0  +  858.0  -  1950.0 ) =  -1534.0
cont( A 2 , A 5 , A 3 ) = 2 * ( 858.0  +  195.0  -  1470.0 ) =  -834.0
cont( A 3 , A 5 , A 4 ) = 2 * ( 195.0  +  195.0  -  900.0 ) =  -1020.0
cont( A 4 , A 5 , A 6 ) = 2 * ( 195.0  +  0.0  -  225.0 ) =  390.0
Here the contribution is maximum value of 650
Hence the order is [A0 A5 A1] is chosen.


Best location for attribute = A6
cont( A 0 , A 6 , A 1 ) = 2 * ( 0  +  0.0  -  0 ) =  0.0
cont( A 1 , A 6 , A 2 ) = 2 * ( 0.0  +  0.0  -  325.0 ) =  -650.0
cont( A 2 , A 6 , A 3 ) = 2 * ( 0.0  +  225.0  -  1950.0 ) =  -3450.0
cont( A 3 , A 6 , A 4 ) = 2 * ( 225.0  +  675.0  -  1470.0 ) =  -1140.0
cont( A 4 , A 6 , A 5 ) = 2 * ( 675.0  +  225.0  -  900.0 ) =  0.0
```

cont(A 5 , A 6 , A 7) = 2 * (225.0 + 0.0 - 0.0) = 450.0
Here the contribution is maximum value of 450
Hence the order is [A5 A6 A7] is chosen.

Best location for attribute = A7
cont(A 0 , A 7 , A 1) = 2 * (0 + 0.0 - 0) = 0.0
cont(A 1 , A 7 , A 2) = 2 * (0.0 + 0.0 - 325.0) = -650.0
cont(A 2 , A 7 , A 3) = 2 * (0.0 + 0.0 - 1950.0) = -3900.0
cont(A 3 , A 7 , A 4) = 2 * (0.0 + 0.0 - 1470.0) = -2940.0
cont(A 4 , A 7 , A 5) = 2 * (0.0 + 0.0 - 900.0) = -1800.0
cont(A 5 , A 7 , A 6) = 2 * (0.0 + 0.0 - 225.0) = -450.0
cont(A 6 , A 7 , A 8) = 2 * (0.0 + 0.0 - 0.0) = 0.0
Here the contribution is maximum value of 0. The 2 orders having same zero value. Any order can be chosen arbitrarily.
Hence the order is [A0 A7 A1] is chosen.

Best location for attribute = A8
cont(A 0 , A 8 , A 1) = 2 * (0 + 0.0 - 0) = 0.0
cont(A 1 , A 8 , A 2) = 2 * (0.0 + 0.0 - 0.0) = 0.0
cont(A 2 , A 8 , A 3) = 2 * (0.0 + 0.0 - 325.0) = -650.0
cont(A 3 , A 8 , A 4) = 2 * (0.0 + 0.0 - 1950.0) = -3900.0
cont(A 4 , A 8 , A 5) = 2 * (0.0 + 0.0 - 1470.0) = -2940.0
cont(A 5 , A 8 , A 6) = 2 * (0.0 + 0.0 - 900.0) = -1800.0
cont(A 6 , A 8 , A 7) = 2 * (0.0 + 0.0 - 225.0) = -450.0
cont(A 7 , A 8 , A 9) = 2 * (0.0 + 0 - 0) = 0.0
Here the contribution is maximum value of 0. The 2 orders having same zero value. Any order can be chosen arbitrarily.
Hence the order is [A0 A8 A1] is chosen.

Hence the final order is [A8, A7, A5, A1, A3, A2, A4, A6]

CA =
```
[[ A8.  A7.  A5.  A1.  A3.  A2.  A4.  A6.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 13.  0. 13.  0.  0.  0.]
 [ 0.  0.  0. 25. 25.  0.  0.  0.]
 [ 0.  0. 13. 25. 53. 15. 15.  0.]
 [ 0.  0.  0.  0. 15. 30. 15. 15.]
 [ 0.  0.  0.  0. 15. 15. 15.  0.]
 [ 0.  0.  0.  0.  0. 15.  0. 15.]]
```

Partitioning:

Z= (CTQ*CBQ)-(COQ*COQ)
Select maximum Z value

```
Fragments = [A8] [A7, A5, A1, A3, A2, A4, A6]
TA = [A8]
TB = [A7, A5, A1, A3, A2, A4, A6]
TQ = []
BQ = [Q2, Q3, Q4, Q8]
OQ = [Q1, Q5, Q6, Q7]
Calculated Z value is:

z = [-6400.0]

Fragments = [A8, A7] [A5, A1, A3, A2, A4, A6]
TA = [A8, A7]
TB = [A5, A1, A3, A2, A4, A6]
TQ = []
BQ = [Q2, Q3, Q4, Q8]
OQ = [Q1, Q5, Q6, Q7]
z = [ -6400.0]

Fragments = [A8, A7, A5] [A1, A3, A2, A4, A6]
TA = [A8, A7, A5]
TB = [A1, A3, A2, A4, A6]
TQ = []
BQ = [Q2, Q3, Q4]
OQ = [Q1, Q5, Q6, Q7,Q 8]
z = [ 0]

Fragments = [A8, A7, A5, A1] [A3, A2, A4, A6]
TA = [A8, A7, A5, A1]
```

TB =[A3, A2, A4, A6]
TQ = []
BQ = [Q2, Q4]
OQ = [Q1, Q3, Q5, Q6, Q7, Q8]
z = [-13924.0]

Fragments = [A8, A7, A5, A1, A3] [A2, A4, A6]
TA = [A8, A7, A5, A1, A3]
TB = [A2, A4, A6]
TQ = [Q3, Q8]
BQ = [Q2]
OQ = [Q1, Q4, Q5, Q6, Q7]
z = [0]
Fragments = [A8, A7, A5, A1, A3, A2] [A4, A6]
TA = [A8, A7, A5, A1, A3, A2]
TB = [A4, A6]
TQ = [Q3, Q8]
BQ = []
OQ = [Q1, Q2, Q4, Q5, Q6, Q7]
z = [-12100.0]
Fragments = [A8, A7, A5, A1, A3, A2, A4] [A6]
TA = [A8, A7, A5, A1, A3, A2, A4]
TB = [A6]
TQ = [Q3, Q4, Q8]
BQ = []
OQ = [Q1, Q2, Q5, Q6, Q7]
z = [-9025.0]

Maximum Z value is 0. So fragmentation is application dependent.so we don’t want to do any fragmentation here.

The Cluster Affinity matrix calculation and partitioning of calculated Cluster Affinity matrix to get fragments using vertical fragmentation should be done similarly as shown above in case of Category, Orders, Vendor, Courier, Payment relations.

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.

Let us assume following are the fragments to be stored:

● Fragment 1

Attribute name	Size
User_id	Int(10)
User_name	Char(20)
User_location	Char(25)
User_email	Char(30)
User_Contact_no	Int(10)

● Fragment 2

Attribute name	Size
P_id	Int(10)
P_name	Char(25)
User_id	Int(10)
Price	Int(6)
Quantity	Int(10)
Manufacture_year	Date(3)
Vendor_id	Int(10)
Cat_id	Int(10)

● Fragment 3

Attribute name	Size
Cat_id	Int(10)
Cat_name	Char(20)

● Fragment 4

Attribute name	Size
Vendor_id	Int(10)
Vendor_name	Char(20)
Vendor_location	Char(25)
Vendor_Contact_no	Int(10)

● Fragement 5

Attribute name	Size
O_id	Int(10)
Co_id	Int(10)
Orderdate	Date(3)
Quantity	Int(10)
Trace_order	Char (20)
User_id	Int(10)
Pay_id	Int(10)
O_id	Int(10)

● Fragment 6

Attribute name	Size
Pay_id	Int(10)
P_id	Int(10)
Pay_mode	Char(15)
Amount_paid	Int()
Discount_given	Int()
Pay_date	Date(3)

● Fragment 7

Attribute name	Size
Co_id	Int(10)
Co_name	Char(20)
Co_contact_no	Int(10)
Co_location	Char(25)
Shipping_address	Char(30)

Delivery_status	Char(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	Total Size(B)	No. of Blocks Size(B)
F1	2058	49	100842	103
F2	614	28	17192	17
F3	1058	34	35972	36
F4	1800	32	57600	57
F5	1026	37	37962	38
F6	300	22	6600	7
F7	2058	24	49392	4

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 = 8B Block Size = 1024B

Fragment	No of Blocks	Size of the index table	Total Size(B)	No. of index Blocks (B)
F1	103	10+8=18	1854	34
F2	17	10+8=18	306	6
F3	36	10+8=18	648	12
F4	57	10+8=18	1026	19
F5	38	10+8=18	684	13
F6	7	10+8=18	126	3
F7	49	10+8=18	882	16

Hence from here we can see that we have reduced the no of block access required for fragment 1 from 103 to 34 using primary indexing. In the worst case we will have to access 35 blocks. 34

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.

8. REPLICATION AND ALLOCATION

Assumptions and List of Formulae:

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

TransmissionDelay (Td) = Packet Size / Bandwidth

Where, Speed of transmission media = 2.7 *

10^6 m/sPacket 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 thenthe 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 isbeing 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
S1	S4	270	100	1
S2	S3	270	100	1
S2	S4	270	100	1
S3	S4	270	100	1

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

Fragment s	**Local retrieval time (ms)**	**Remote retrieval time (ms)**	**Local update time (ms)**	**Remote update time (ms)**
F1	4845.1 5	5046.1 5	9690.3	9990.3
F2	174.6	474.6	349.2	649.2
F3	1571.4	1872.4	3142.8	3442.8
F4	3201	3501	6402	6702
F5	9952.2	10252. 2	19904.4	20204.4
F6	778.2	1078.2	1990.4	2290.4
F7	778	1078	692	992

For Allocation, we are using Redundant All Beneficial Site

Method.Assume the following:

Transactions(Queries)	Site	Frequency	Fragment Access
Q1	S1	50	F1 1Read
Q2	S3	40	F2 2Read
Q3	S1	30	F1 2Read
Q4	S4	50	F3 2Read F8 1Read
Q5	S1,S2	50,40	F6 2Read F4 1Read F5 3 Read
Q6	S2,S3	50,30	F5 2Read F2 3Read F7 2Read
Q7	S3,S4,S2	50,20,30	F7 1Read F5 2Read F2 1 Read F3 3Read
Q8	S1,S4,S3	50,30,40	F4 1Read F3 2Read F1 2 Read F2 1Read

Since our sample queries are related only read (not update) will calculate only Benefit Computation (not Cost computation) of placing a fragment at a particular site. Let us proceed to Benefit Computation. Benefit computation is based on read queries. The benefit of placing each fragment at each site is given in the below table.

Fragment	**Site**	**Queries**	**#read*frequency*(remote-Local time)**	**Benefit**	**Benefit-Cost**
F1	S1	Q1,Q3,Q8	1*50*300+2*30*300+ 2*50*300	63000	63000
	S2	-	-	0	0
	S3	Q8	2*50*300	30000	30000
	S4	Q8	2*50*300	30000	30000
F2	S1	Q8	1*50*300	15000	15000
	S2	Q7,Q6	1*30*300+3*50*300	54000	54000
	S3	Q2,Q6,Q7, Q8	2*40*300+3*30*300+ 1*30*300+1*40*300	72000	72000
	S4	Q7,Q8	1*20*300+3*30*300	33000	33000
F3	S1	Q8	2*50*300	30000	30000
	S2	Q7	30*3*300	27000	27000
	S3	Q8,Q7	2*40*300+ 3*50*300	69000	69000
	S4	Q4	2*50*300	30000	30000
F4	S1	Q5,Q8	1*50*300+1*50*300	30000	30000
	S2	Q5	1*40*300	12000	12000
	S3	Q8	1*40*300	12000	12000

	S4	Q8	1*30*300	9000	9000
F5	S1	Q5	3*50*300	45000	45000
	S2	Q5,Q6,Q7	3*40*300+2*50*300 +2*30*300	84000	84000
	S3	Q6,Q7	2*30*300+2*50*300	48000	48000
	S4	Q7	2*20*300	12000	12000
F6	S1	Q5	2*50*300	30000	30000
	S2	Q5	2*40*300	24000	24000
	S3	-	-	0	0
	S4	-	-	0	0
F7	S1	-	-	0	0
	S2	Q6,Q7	2*50*300+1*30*300	39000	39000
	S3	Q6,Q7	2*30*300+1*50*300	33000	33000
	S4	Q7	1*20*300	6000	6000

Allocation:

Site	**Fragments**
S1	F1,F2,F3,F4,F5,F6
S2	F2,F3,F4,F5,F6,F7
S3	F1,F2,F3,F4,F5,F7
S4	F1F3,F4,F5,F7

Work Area Space and System Specification

Total Disk Space required is = 100842 + 17192 + 35972 + 57600 + 37962 + 6600 + 49392 + 33600 = 339.16 KB. Assumed network speed = 1 MBps. Max no of records in our fragment are 2058 B and the maximum size is 49 B .Total Buffer size required is (Assume at a time we need one fragment) = 2058*49 = 101 KB