NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

Department of Information Technology

Advanced Database Systems

Assignment 1

STUDENT INFORMATION SYSTEM

Submitted by

Name : K.SHIVA KUMAR

Roll No: 222IT013

Class : M.Tech, I-semester

Contents

1. PROBLEM DESCRIPTION

The student information system (SIS) used to manage the information related to students. In this assignment, we are mainly focused on the design of the database for maintaining the information at four independent data sources that are connected logically.

Mr. John has many institutes, many Hostels, a Library, and a Placement office. He wants to connect them together by focusing on the student information. Admissions and academics for the institutes happen at one location. The different activities performed at each place are described below

Academics:

In the admission section, they will keep the information about the student like name, roll number, date of birth, gender, year enrolled, phone number, program, specialization, CGPA.

There they also store the information about the fees student paid with the name of the student, roll number, Acadamic_fee,fee_reciept_number, status,

In the academic section, they will keep the information about the course's student registered. one course can be registered by many students. one student can register with many courses based on the courses available in the semester. They will store the information about the courses like the course name, course id, roll number, credits, grade, semester

Hostel admission office:

Mr. John also has some hostels which provide accommodation for the students who are studying in his institutes. To maintain the admission at these hostels Mr. John has a hostel office where the admission will happen. Each hostel will be allocated to students based on the institute they are studying, year of study, the degree they are pursuing like all engineering students will be accommodated at one hostel wing, all medical students will be accommodated at one hostel wing, etc.

It will also accommodate other people like staff, faculty, guests at the guest house. Each hostel will have a mess. Each hostel mess is supervised by the hostel admission office.

If a student wants accommodation in the hostel he/she has to show the admit card given by the admission office and student has to give the identity proofs to the hostel admission office. He has to pay the fee based on the type of the room he wanted to stay. Hostel admission office maintains the record for each student for the fee payments. If a student didn't keep the amenities provided by the hostel he/she will be fined. Student will get a mess identification number in which he wants to take food.

In the Hostel admission office, they will keep the information about the student who is staying in the hostel and mess identification number, mess name, name of the hostel, room number, the fee paid, and fine. One student can get accommodation in one room.

Training and Placement Cell:

Training and Placement cell deals with inviting companies to hire students who are studying in Mr. John institutes.

In general Training and Placement cell team will invite the companies to its office to offer placements and internships for the students. After having the meeting with recruiters training and placement cell will publish the details of the recruitment.

If a student wants to apply for Placement or internships he/she has to register for the recruitment at training and placements cell. If a student is eligible for the recruitment they will forward the application to the recruiters. After the recruitment process completed Training and placement cell release the results. It will maintain the record of students who are hired.

In the Training and Placement Cell, they will store the information about the student name, placed company, CTC (in lakh), internship or Placement, Job role. One student can get placed in only one company. One company can hire many students.

Library:

Mr. John has a Library that will provide the books to the students who are studying in his institutes. This library contains books related to engineering, medical and Law. It will also provide books to the faculty, staff.

The library office maintains the details of the books available. It will maintain the details of the student who took the book. The library office has to deal with the vendors who are selling the books.

If a student wants to take a book from the library. He has to take the library membership card from the library by paying fee. Student has to login to the portal provided by the library and search for the availability of the book in the catalog, if the book found available student can request for the book and go to the library and take the book. If the student keeps the textbooks for more than 15 days or the book taken is not properly returned then the student will have to pay the fine. The fine has to pay to the library office.

In the library office, they will keep the information about the book name, book id, student name who taken the book, roll number of the student, date of borrow, date of return, status(book returned or not).

And they also store about the fine student has to pay.

However, all these information are not maintained in a single location because different department maintains its own information system. Therefore, a distributed scenario comes into a picture that allows control and data flow between multiple data sources to share information. Based on the requirement, four data sources are identified. The data sources are

1. Academics
2. Library
3. Hostel admission office
4. Training and placement office

Actors: people who interact with the database

1. Administrator
2. Librarian

Queries:

1. Find all the student names who are joined in the year 2021

2. Find all the student names who are staying in the hostel “X”

3. Find all the student names who passed in the course “A”

4. Find all the student names who hired with more than 10 lakh package

5. Find the student who registered to the course “A” and taken a book “B” from the library

6. Find the student who has to pay fine in the both Library section and Hostel section

7. Find the name of the student who is staying in hostel X, hired by a company with more than 10 lakh package, and has fine in the library

8. Find the names of students who have a CGPA>6.0, hired with more than 10 lakhs by a company, staying in hostel “X”

2. ENTITIES THAT ARE MAINTAINED IN DIFFERENT LOCATIONS

ENTITIES	LOCATION	INFORMATION
student,coursesregistered,fees	Site 1	Gives the information about the student, courses registered, the fee paid by the
Book_information	Site 2	Gives the information about who took the book, return date, fine
HostelRoom	Site 3	Gives information about the hostel room
PlacementDetails,company details	Site 4	Gives information about placement Details, like CTC, company name, job title

ENTITY SETS:

Placement details(Placement id,company id,student name,CTC,recruitementtype)

CoursesRegistered(Course_name,Course_id,credits,grade,semester)

Fees(name,academic_fee,fee_reciept_number,status)

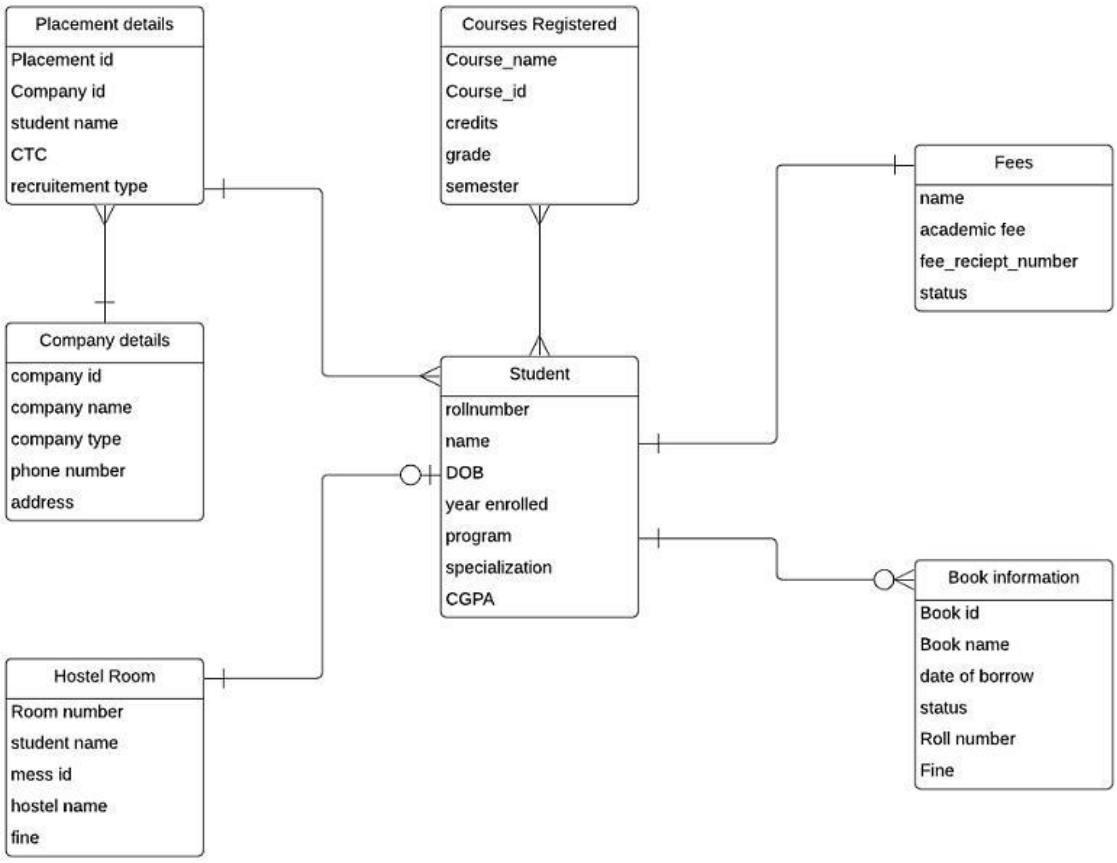
Student(roll number,name,DOB,year enrolled,program,specialization,CGPA)

Companydetails(companyid,companyname,company type,phone number,address)

Hostel Room(Room number,student name,mess id,hostelname,fine)

Book_Information(bookid,bookname,dateofborrow,status,rollnumber,fine)

3. EER MODEL


Placement details
Placement id
Company id
student name
CTC
recruitement type
Courses Registered
Course_name
Course_id
credits
grade
semester
Fees
name
academic fee
fee_reciept_number
status
Company details
company id
company name
company type
phone number
address
Student
rollnumber
name
DOB
year enrolled
program
specialization
CGPA
Book information
Book id
Book name
date of borrow
status
Roll number
Fine
Hostel Room
Room number
student name
mess id
hostel name
fine

4. GLOBAL CONCEPTUAL SCHEMA

The transformation from the entity-relationship model to the relational model is very straightforward. A feasible set of relational schemas is as follows

STUDENT:

Roll number	Student name	DOB	Year enrolled	program	specialization	CGPA	Placement Id

HOSTEL ROOM:

Room number	Student name	Mess id	Hostel name	fine	Roll number

Placement details:

Placement id	Student name	CTC	Recruitment type	Company name

Company details:

Company id	Company name	Company type	Phone number	city

Courses Registered:

Course name	Course name	credits	Grade	semester	Roll number

Fees:

Student name	Academic fee	Fee_reciept_no	Roll number

Book information:

Book id	Book name	Date of barrow	status	fine	Roll number

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. 1^{st} 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.

A → BC

A	B	C
1	c	d
2	a	b
3	e	f

→This symbols denotes "determines" or "returns"

(group of attributes) → (group of attributes)

LHS is called key and RHS is called values.

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

   Example: A determines itself.

   $A \rightarrow A$

   $A \rightarrow AB$

   $AB \rightarrow A$

   In general form to represent this

   $X \supseteq Y$

   however, there are no useful operation using this kind of dependency.

2. **Non-trivial FD:** Given a value of an attribute get a unique

   value.

   Example:

   $A \rightarrow B$

   $A \rightarrow BC$

   $AB \rightarrow CD$

   In general, the rule is

   $X \cap Y = \varphi$

3. **Semi non-trivial FD:** it is not deriving completely new information. It partially brings a record, that is, not the value of entire set of attributes.

Example:

AB→BC B is common on both sides

ACD→ EFCD CD is common both sides

It is generally represented as

$X \cap Y \neq \varphi$

FD's are quite useful in identifying keys, identifying equivalences of FS's and finding minimal FD set. There are rules applied on FD such as inference rules (reflexive, transitivity, augmentation, union, etc), closure.

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

$2_n - 1$

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

1st 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.

STUDENT:

Roll number	Student name	DOB	Year enrolled	program	specialization	CGPA	Placement Id

HOSTEL ROOM:

Room number	Student name	Mess id	Hostel name	fine	Roll number

Placement details:

Placement id	Student name	CTC	Recruitment type	Company name

Company details:

Company id	Company name	Company type	Phone number	address

Courses Registered:

Course id	Course name	credits	grade	semester	Rollnumber

Fees:

Fee_reciept_no	Academic fee	Student name	Roll number

Book information:

Book id	Book name	Date of barrow	status	fine	Roll number

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:

- ✓ Repeating column values are taken out and maintained in a separate table. So that change can be done only once in the new table rather than all entries in the first table. Rule is foreign key must be on the N side else again multi-value in a column will occur.
- ✓ Identify **prime attribute** (part of candidate key that determines anything else), it is also called **partial dependency**, and eliminate it. Because, 2nd NF is based on **Full Functional dependency** (key should determine all other attributes in a table)
- ✓ Use foreign key on many side

STUDENT:

Roll number	Student name	DOB	Year enrolled	program	specilisation	CGPA	Placement Id

HOSTEL ROOM:

Room number	Student name	Mess id	Hostel name	fine	Roll number

Placement details:

Placement id	Student name	CTC	Recruitement type	Company name

Company details:

Company id	Company name	Company type	Phone number	city

Courses Registered:

Course id	Course name	credits	grade	semester	Rollnumber

Fees:

Fee_reciept_no	Academic fee	Student name	Roll number

Book information:

Book id	Book name	Date of barrow	status	fine	Roll number

3rd Normal Form:

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.

3rd 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.

STUDENT:

Roll number	Student name	DOB	Year enrolled	program	Specilisation	CGPA	Placement Id

HOSTEL ROOM:

Room number	Student name	Mess id	Hostel name	Fine	Roll number

Placement details:

Placement id	Student name	CTC	Recruitement type	Company name

Company details:

Company id	Company name	Company type	Phone number	city

Courses Registered:

Course id	Course name	credits	grade	semester	Rollnumber

Course Id	Course name	Credits
IT700	Algorithms	4
IT701	Advanced Database	4
IT702	Deep learning	4

Fees:

Fee_reciept_no	Academic fee	Student name	Roll number

Book information:

Book id	Book name	Date of barrow	status	Fine	Roll number

Every candidate key in a table determines all other attributes and no non-key attributes determine any attributes in the tables.

4^{th} NF is not required since we eliminate repeated values in the 2NF itself.

BCNF:

Every 3NF is not BCNF but if a table is in BCNF then it is already in 3NF. **BCNF** says every LHS of FD's must be the key of one of the tables. That is, every prime attribute should determine all other attributes in a table. Therefore, the above tables satisfy BCNF.

6. GLOBAL SCHEMA

STUDENT	
Attribute name	**Attribute size(type in bytes)**
Roll number	Char(10)
Student name	Char(10)
DOB	Date
Year Enrolled	Int(2)
Program	Char(10)
Specialization	Char(10)
CGPA	Float(2)
Placement Id	Int(2)

HOSTEL_ROOM	
Attribute name	**Attribute size(type in bytes)**
Room number	Int(2)
Student name	Char(10)
Mess id	Int(2)
Hostel name	Char(10)
Fine	Int(2)
Roll number	Char(2)

PLACEMENT_DETAILS	
Attribute name	**Attribute size(type in bytes)**
Placement id	Int(2)
Student name	Char(10)
CTC	Int(2)
Recruitment type	Char(10)
Company name	Char(10)

COMPANY_DETAILS	
Attribute name	**Attribute size(type in bytes)**
Company id	Int(2)
Company name	Char(10)
Company type	Char(10)
Phone number	Char(10)
Address	Char(10)

Courses	
Attribute name	**Attribute size(type in bytes)**
Course id	Char(10)
Course name	Char(10)
Credits	Int(2)

COURSES_REGISTERED	
Attribute name	**Attribute size(type in bytes)**
Course id	Char(10)
Grade	Int(2)
Semester	Char(10)
Roll number	Char(10)

Fees:	
Attribute name	**Attribute size(type in bytes)**
Student name	Char(10)
Academic fee	Int(2)
Fee_reciept_no	Int(2)
Roll number	Char(10)

BOOK_INFORMATION:	
Attribute name	**Attribute size(type in bytes)**
Book id	Int(2)
Book name	Char(10)
Date of Barrow	Date
Status	Char(10)
Fine	Int(10)
Roll number	Char(2)

7. FRAGMENTATION

Database tables are usually decomposed into smaller fragments for following reasons:

1. when storage exhausted out
2. for parallel processing
3. for load balancing
4. to improve query response time
5. for better local processing
6. availability

Decomposed fragments are placed into some other site to facilitate query and optimize other quality of services. These fragments permit number of transactions concurrently. Taking copy of a relation and maintaining 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:

(i) **Completeness**: all row or column must be present in at least one site.

(ii) **Reconstruction:** while reconstructing the relation, there should not be any inconsistency or loss of data.

(iii) **Disjointness:** 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 student information system.

1. Find all the student names who are joined in the year 2021

```
SELECT STUDENTNAME
FROM STUDENT
WHERE YEAR_ENROLLED=2021
```

2. Find all the student names who are staying in the hostel X

```
SELECT STUDENTNAME
FROM HOSTELROOM
WHERE HOSTELNAME=X
```

3. Find all the student names who passed in the course “A”

```
SELECT STUDENTNAME

FROM COURSE_REGISTERED, STUDENT

WHERE COURSE_REGISTERED.ROLLNUMBER=STUDENT.ROLLNUMBER

COURSE_REGISTERED.GRADE>4.0 AND STUDEN.COURSE_ID=A
```

4. Find all the student names who hired with more than 10 lakh package

```
SELECT STUDENTNAME

FROM PLACEMENT_DETAILS

WHERE CTC>10
```

5. Find the student who registered to the registered to the course “A” and taken a Book “B” from the library

```
SELECT STUDENTNAME

FROM COURSES_REGISTERED, BOOK_INFORMATION

WHERE
COURSE_REGISTERED.ROLLNUMBER=BOOK_INFORMATION.ROLLNUMBER
AND COURSE_REGISTERED.COURSE_NAME=A AND
BOOK_INFORMATION.BOOK_NAME=B
```

6. Find the student who has to pay fine in the both Library section and Hostel section

```
SELECT STUDENTNAME

FROM BOOK_INFORMATION, HOSTELROOM

WHERE HOSTELROOM.FINE>0 AND BOOK_INFORMATION.FINE>0
```

7. Find the names of the student who is staying in hostel X, hired by a company with more than 10 lakh package ,and has fine in the library

```
SELECT STUDENTNAME
FROM HOSTELROOM, PLACEMENT_DETAILS, BOOK_INFORMATION
WHERE HOSTEL.ROLLNUMBER=PLACEMENT_DETAILS.ROLLNUMBER
AND HOSTELROOM.ROLLNUMBER=BOOK_INFORMATION.ROLLNUMBER
AND HOSTELROOM.HOSTELNAME=X AND PLACEMENT_DETAILS.CTC>10
AND BOOK_INFORMATION.FINE>0
```

8. Find the names of students who have a CGPA>6.0 hire with more than 10 lakhs by a Company, staying in hostel "X"

```
SELECT STUDENT_NAME
FROM STUDENT, PLACEMENT_DETAILS, HOSTELROOM
WHERE STUDENT.CGPA>6.0 AND
HOSTEL.HOSTEL_NAME="X"  AND
PLACEMENT.CTC>10;
```

7.1. 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 students who are joined in the year 2021.

7.2. Vertical Fragmentation:

The vertical fragmentation of a relation R produces subschema's R1, R2, R2,…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:

S1= admission and academics

S2= library

S3= Hostel admission office

S4= Training and placement office

Frequency Matrix:

Assume frequency matrix as follow:

Sites **Queries**	**S1**	**S2**	**S3**	**S4**	**Total queries**
Q1	30	0	0	0	30
Q2	0	0	30	0	30
Q3	30	0	0	0	30
Q4	0	0	0	30	30
Q5	30	30	0	0	60
Q6	0	30	30	0	60
Q7	30	0	0	30	60
Q8	0	30	30	30	90

Relation-1: STUDENT

Attribute Usage Matrix:

Attributes Queries	A1 Roll number	A2 Student name	A3 DOB	A4 Year Enrolled	A5 Program	A6 Specialization	A7 CGPA	A8 Placement Id
Q1	0	1	0	1	0	0	0	0
Q2	0	0	0	0	0	0	0	0
Q3	1	0	0	0	0	0	0	0
Q4	0	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	0	0	0	0
Q8	0	1	0	0	0	0	1	0

Attribute Affinity Matrix:

	A1	A2	A3	A4	A5	A6	A7	A8
A1	30	0	0	0	0	0	0	0
A2	0	120	0	30	0	0	0	0
A3	0	0	0	0	0	0	0	0
A4	0	30	0	30	0	0	0	0
A5	0	0	0	0	0	0	0	0
A6	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	90	0
A8	0	0	0	0	0	0	0	0

To calculate the Cluster Affinity matrix ordering we make use of the Bond Energy Algorithm (BEA). BEA groups the attributes which have more affinity and groups which have less affinity. For this we make use of following functions.

$Cont(A_i,A_j,A_k)$: $2*[Bond(A_i,A_j) + Bond(A_j,A_k) - Bond(A_i,A_k)]$

$Bond(A_i,A_j)$: $\sum^{n}_{p=1} AA(A_p,A_i) * AA(A_p,A_j)$

Ordering the position of 3rd column A3 of AA:

1. Cont(0,3,1)=2*(Bond(0,3)+Bond(3,1)-Bond(0,1))=2*(0+0-0)=0
2. Cont(1,3,2)=2*(Bond(1,3)+Bond(3,2)-Bond(1,2))=2*(0+0-0)=0
3. Cont(2,3,4)=2*(Bond(2,3)+Bond(3,4)-Bond(2,4))=2*(0+0-4500)=-9000

Maximum value of Contribution is 0. Multiple functions producing the same value. Hence, we can choose any position.Now order is CA(1,3,2)

Ordering the position of 4th column A4 of AA:

1. Cont(0,4,1)=2*(Bond(0,4)+Bond(4,1)-Bond(0,1))=2(0+0-0)=0
2. Cont(1,4,3)=2*(Bond(1,4)+Bond(4,3)-Bond(1,3))=2(0+0-0)=0
3. Cont(3,4,2)=2*(Bond(3,4)+Bond(4,2)-Bond(2,3))=2(0+4500-0)=9000
4. Cont(2,4,5)=2*(Bond(2,4)+Bond(4,5)-Bond(2,5))=2(4500+0-0)=9000

Maximum value of Contribution is 9000. Multiple functions producing the same value. Hence, we can choose any position. Now order is CA(1,3,2,4)

Ordering the position of 5th column A5 of AA:

1. Cont(0,5,1)=2*(Bond(0,5)+Bond(5,1)-Bond(0,1))=2(0+0-0)=0
2. Cont(1,5,3)=2*(Bond(1,5)+Bond(5,3)-Bond(1,3))=2(0+0-0)=0
3. Cont(3,5,2)=2*(Bond(3,5)+Bond(5,2)-Bond(2,3))=2(0+0-0)=0
4. Cont(2,5,4)=2*(Bond(2,5)+Bond(5,4)-Bond(2,4))=2(0+0-4500)=-9000
5. Cont(4,5,6)=2*(Bond(4,5)+Bond(5,6)-Bond(4,6))=2(0+0-0)=0

Maximum value of Contribution is 0. Multiple functions producing the same value. Hence, we can choose any position. Now order is CA(1,3,2,4,5)

Ordering the position of 6th column A6 of AA:

1. Cont(0,6,1)=2*(Bond(0,6)+Bond(6,1)-Bond(0,1))=2(0+0-0)=0
2. Cont(1,6,3)=2*(Bond(1,6)+Bond(6,2)-Bond(1,2))=2(0+0-0)=0
3. Cont(3,6,2)=2*(Bond(3,6)+Bond(6,2)-Bond(2,3))=2(0+0-0)=0
4. Cont(2,6,4)=2*(Bond(2,6)+Bond(6,4)-Bond(2,4))=2(0+0-4500)= -9000
5. Cont(4,6,5)=2*(Bond(4,6)+Bond(6,5)-Bond(4,5))=2(0+0-0)=0
6. Cont(5,6,7)=2*(Bond(5,6)+Bond(6,7)-Bond(5,7))=2(0+0-0)=0

Maximum value of Contribution is 0. Multiple functions are producing the same value. We can choose any one. Now order is CA(1,3,2,4,5,6)

Ordering the position of 7th column A7 of AA:

1. Cont(0,7,1)=2*(Bond(0,7)+Bond(7,1)-Bond(0,1))=2(0+0-0)=0
2. Cont(1,7,3)=2*(Bond(1,7)+Bond(7,2)-Bond(1,3))=2(0+0-0)=0
3. Cont(3,7,2)=2*(Bond(2,7)+Bond(7,3)-Bond(2,3))=2(0+0-0)=0
4. Cont(2,7,4)=2*(Bond(3,7)+Bond(7,4)-Bond(2,4))=2(0+0-4500)= -9000
5. Cont(4,7,5)=2*(Bond(4,7)+Bond(7,5)-Bond(4,5))=2(0+0-0)=0
6. Cont(5,7,6)=2*(Bond(5,7)+Bond(7,6)-Bond(5,6))=2(0+0-0)=0
7. Cont(6,7,8)=2*(Bond(6,7)+Bond(7,8)-Bond(6,8))=2(0+0-0)=0

Maximum value of Contribution is 0. Multiple functions are producing the same value. We can choose any one. Now order is CA(1,3,2,4,5,6,7)

Ordering the position of 8th column A8 of AA:

1. Cont(0,8,1)=2*(Bond(0,8)+Bond(8,1)-Bond(0,1))=2(0+0-0)=0
2. Cont(1,8,3)=2*(Bond(1,8)+Bond(8,3)-Bond(1,3))=2(0+0-0)=0
3. Cont(3,8,2)=2*(Bond(2,8)+Bond(8,3)-Bond(2,3))=2(0+0-0)=0
4. Cont(2,8,4)=2*(Bond(3,8)+Bond(8,4)-Bond(2,4))=2(0+0-4500)=-9000
5. Cont(4,8,5)=2*(Bond(4,8)+Bond(8,5)-Bond(4,5))=2(0+0-0)=0
6. Cont(5,8,6)=2*(Bond(5,8)+Bond(8,6)-Bond(5,6))=2(0+0-0)=0
7. Cont(6,8,7)=2*(Bond(6,8)+Bond(7,8)-Bond(6,7))=2(0+0-0)=0
8. Cont(7,8,9)=2*(Bond(7,8)+Bond(8,9)-Bond(7,9))=2*(0+0-0)=0

Maximum value of Contribution is 0. Multiple functions are producing the same value. We can choose any one. Now order is CA(1,3,2,4,5,6,7,8)

Clustering Affinity Matrix:

	A1	A3	A2	A4	A5	A6	A7	A8
A1	30	0	0	0	0	0	0	0
A3	0	0	0	0	0	0	0	0
A2	0	0	120	30	0	0	0	0
A4	0	0	30	30	0	0	0	0
A5	0	0	0	0	0	0	0	0

A6	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	90	0
A8	0	0	0	0	0	0	0	0

Partitioning is based on following metrics:

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

Partitioning of Cluster affinity matrix:

TA={A1} TB={A3,A2,A4,A5,A6,A7,A8}

TQ={Q3} BQ={Q1,Q8} OQ={}

CTQ=30 CBQ=120 COQ=0

Z=30*120-0=3600

TA={A1,A3} TB={A2,A4,A5,A6,A7,A8}

TQ={Q3} BQ={Q1,Q8} OQ={}

CTQ=30 CBQ=120 COQ=0

Z=30*120-0=3600

TA={A1,A3,A2} TB={A4,A5,A6,A7,A8}

TQ={Q3} BQ={} OQ={Q1,Q8}

CTQ=30 CBQ=0 COQ=120

Z=30*0-120*120= -14400

TA={A1,A3,A2,A4} TB={A5,A6,A7,A8}

TQ={Q1,Q3} BQ={} OQ={Q8}

CTQ=60 CBQ=0 COQ=90

Z=60*0-90*90= -8100

TA={A1,A3,A2,A4,A5} TB={A6,A7,A8}

TQ={Q1,Q3} BQ={} OQ={Q8}

CTQ=60 CBQ=0 COQ=90

Z=60*0-90*90= -8100

TA={A1,A3,A2,A4,A5,A6} TB={A7,A8}

TQ={Q1,Q3} BQ={} OQ={Q8}

CTQ=60 CBQ=0 COQ=90

Z=60*0-90*90= -8100

TA={A1,A3,A2,A4,A5,A6,A7} TB={A8}

TQ={Q1,Q3,Q8} BQ={} OQ={}

CTQ=150 CBQ=0 COQ=0

Z=150*0-0=0

Most of the values of z are 0.so fragmentation is application dependent. so we don't want to do any fragmentation here.

Relation-2: HOSTELROOM

Attribute Usage Matrix:

Attributes	A1 Room number	A2 Student name	A3 Mess id	A4 Hostel name	A5 Fine	A6 Roll number
Q1	0	0	0	0	0	0
Q2	0	1	0	1	0	0
Q3	0	0	0	0	0	0
Q4	0	0	0	0	0	0
Q5	0	0	0	0	0	0
Q6	0	1	0	0	1	0
Q7	0	1	0	1	0	1
Q8	0	1	0	1	0	0

Attribute Affinity Matrix:

	A1	A2	A3	A4	A5	A6
A1	0	0	0	0	0	0
A2	0	240	0	180	60	60
A3	0	0	0	0	0	0
A4	0	180	0	180	60	60
A5	0	60	0	60	60	0
A6	0	60	0	60	0	60

Ordering the position of 3rd column A3 in AA:

1. Cont(0,3,1)=2*(Bond(0,3)+ Bond(3,1)- Bond(0,1))=2(0+0-0)=0
2. Cont(1,3,2)=2*(Bond(1,3)+ Bond(3,2)- Bond(1,2))=2(0+0-0)=0
3. Cont(2,3,4)=2*(Bond(2,3)+ Bond(3,4)- Bond(2,4))=2(0+0-82800)= -82800

Maximum value of Contribution is 0. Multiple functions producing the same value. Hence, we can choose any position. Now order is CA(1,3,2)

Ordering the position of 4^{th} column of A4 in AA:

1. Cont(0,4,1)=2*(Bond(0,4)+Bond(4,1)-Bond(0,1))=2(0+0-0)=0
2. Cont(1,4,3)=2*(Bond(1,4)+Bond(4,3)-Bond(1,3))=2(0+0-0)=0
3. Cont(3,4,2)=2*(Bond(3,4)+Bond(4,2)-Bond(2,3))=2(0+82800-0)=165600
4. Cont(2,4,5)=2*(Bond(2,4)+Bond(4,5)-Bond(2,5))=2(82800+25200-72800)=158400

Maximum value of Contribution is 165600.Hence the order is CA(1,3,4,2)

Ordering the position of 5th column A5 of AA:

1. Cont(0,5,1)=2*(Bond(0,5)+Bond(5,1)-Bond(0,1))=2(0+0-0)=0
2. Cont(1,5,3)=2*(Bond(1,5)+Bond(5,3)-Bond(1,3))=2(0+0-0)=0
3. Cont(3,5,4)=2*(Bond(3,5)+Bond(5,4)-Bond(3,4))=2(0+25200-0)=50400
4. Cont(4,5,2)=2*(Bond(4,5)+Bond(5,2)-Bond(2,4))=2(25200+72800-82800)= 30400
5. Cont(2,5,6)=2*(Bond(2,5)+Bond(5,6)-Bond(2,6))=2(82800+7200-28800)=122400

Maximum value of Contribution is 122400.Hence the order is CA(1,3,4,2,5)

Ordering the position of 6th column A6 in AA:

1. Cont(0,6,1)=2*(Bond(0,6)+ Bond(6,1)- Bond(0,1))=2(0+0-0)=0
2. Cont(1,6,3)=2*(Bond(1,6)+ Bond(6,3)- Bond(1,3))=2(0+0-0)=0
3. Cont(3,6,4)=2*(Bond(3,6)+ Bond(6,4)- Bond(3,4))=2(0+25200-0)=50400
4. Cont(4,6,2)=2*(Bond(4,6)+ Bond(6,2)- Bond(2,4))=2(25200+28800-82800)=28800
5. Cont(2,6,5)=2*(Bond(2,6)+ Bond(6,5)- Bond(2,5))=2(28800+7200-28800)=14400
6. Cont(5,6,7)=2*(Bond(5,6)+ Bond(6,7)- Bond(5,7))=2(7200+0-0)=14400

Maximum value of Contribution is 50400. Hence, Now order of CA(1,3,6,4,2,5)

Clustering Affinity Matrix:

	A1	A3	A6	A4	A2	A5
A1	0	0	0	0	0	0
A3	0	0	0	0	0	0
A6	0	0	60	60	60	0
A4	0	0	60	180	180	60
A2	0	0	60	180	240	60
A5	0	0	0	60	60	60

Partitioning of cluster affinity matrix:

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

TQ={} BQ={Q2,Q6,Q7,Q8} OQ={}

CTQ=0 CBQ=240 COQ=0

Z=0*240-0=0

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

TQ={} BQ={Q2,Q6,Q7,Q8} OQ={}

CTQ=0 CBQ=240 COQ=0

Z=0*240-0=0

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

TQ={} BQ={Q2,Q6,Q8} OQ={Q7}

CTQ=0 CBQ=180 COQ=60

Z=0*180 -60*60=-3600

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

TQ={} BQ={} OQ={Q2,Q6,Q7,Q8}

CTQ=0 CBQ=0 COQ=240

Z=0-240*240= -57600

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

TQ={Q2,Q7,Q8} BQ={} OQ={Q6}

CTQ= 180 CBQ= 0 COQ=60

Z=180*0-60*60= -3600

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

Relation-3:PLACEMENT_DETAILS

Attribute Usage Matrix:

	A1 Placement id	A2 Student name	A3 CTC	A4 Recruitment type	A5 Company name
Q1	0	0	0	0	0
Q2	0	0	0	0	0
Q3	0	0	0	0	0
Q4	0	1	1	0	0
Q5	0	0	0	0	0
Q6	0	0	0	0	0
Q7	0	1	1	0	0
Q8	0	1	1	0	0

Attribute Usage Matrix:

	A1	A2	A3	A4	A5
A1	0	0	0	0	0
A2	0	180	180	0	0
A3	0	180	180	0	0
A4	0	0	0	0	0
A5	0	0	0	0	0

Ordering the position of 3rd column A3 in AA:

1.Cont(0,3,1)=2*(Bond(0,3)+ Bond(3,1)- Bond(0,1))=2(0+0-0)=0

2.Cont(1,3,2)=2*(Bond(1,3)+ Bond(3,2)- Bond(1,2))=2(0+64800-0)=129600

3.Cont(2,3,4)=2*(Bond(2,3)+ Bond(3,4)- Bond(2,4))=2(64800+0-0)=129600

Maximum value of Contribution is 129600. Multiple functions producing the same value. Hence, we can choose any position. Now order is CA(1,2.3)

Ordering the position of 4^{th} column of A4 in AA:

1.Cont(0,4,1)=2*(Bond(0,4)+Bond(4,1)-Bond(0,1))=2(0+0-0)=0

2.Cont(1,4,2)=2*(Bond(1,4)+Bond(4,2)-Bond(1,2))=2(0+0-0)=0

3.Cont(2,4,3)=2*(Bond(2,4)+Bond(4,3)-Bond(2,3))=2(0+0-64800)= -129600

4.Cont(3,4,5)=2*(Bond(3,4)+Bond(4,5)-Bond(3,5))=2(0+0-

0)=0 Maximum value of Contribution is 0.Hence the order is

CA(1,2,3,4) Ordering the position of 5th column A5 of AA:

1.Cont(0,5,1)=2*(Bond(0,5)+Bond(5,1)-Bond(0,1))=2(0+0-0)=0

2.Cont(1,5,2)=2*(Bond(1,5)+Bond(5,2)-Bond(1,2))=2(0+0-0)=0

3.Cont(2,5,3)=2*(Bond(2,5)+Bond(5,3)-Bond(2,3))=2(0+0-64800)= -129600

4.Cont(3,5,4)=2*(Bond(3,5)+Bond(5,4)-Bond(3,4))=2(0+0-0)= 0

5.Cont(4,5,6)=2*(Bond(4,5)+Bond(5,6)-Bond(4,6))=2(0+0-0)=0

Maximum value of Contribution is 0.Hence the order is CA(1,2,3,4,5)

Clustered affinity matrix:

	A1	A2	A3	A4	A5
A1	0	0	0	0	0
A2	0	180	180	0	0
A3	0	180	180	0	0
A4	0	0	0	0	0
A5	0	0	0	0	0

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

TQ={} BQ={Q4,Q7,Q8} OQ={}

CTQ=0 CBQ=180 COQ=0

Z=0*180-0=0

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

TQ={} BQ={} OQ={Q4,Q7,Q8}

CTQ=0 CBQ=0 COQ=180

Z=0*0 -180*180=-32400

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

TQ={Q4,Q7,Q8} BQ={} OQ={}

CTQ=180 CBQ=0 COQ=0

Z=180*0-0=0

For other partitions also Z value becomes 0 and the maximum z value is 0. so fragmentation is application dependent. So we don't want to do any fragmentation here.

Relation-4:BOOK_INFORMATION

Attribute Usage Matrix:

	A1 Book id	A2 Book name	A3 Date of barrow	A4 Status	A5 Fine	A6 Roll number
Q1	0	0	0	0	0	0
Q2	0	0	0	0	0	0
Q3	0	0	0	0	0	0
Q4	0	0	0	0	0	0
Q5	0	1	0	0	0	1
Q6	0	0	0	0	1	0
Q7	0	0	0	0	1	1
Q8	0	0	0	0	0	0

Attribute Affinity Matrix:

	A1	A2	A3	A4	A5	A6
A1	0	0	0	0	0	0
A2	0	60	0	0	0	60
A3	0	0	0	0	0	0
A4	0	0	0	0	0	0
A5	0	0	0	0	120	60
A6	0	60	0	0	60	60

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 STUDENT AND HOSTELROOM relations.

Relation-5:COURSES_REGISTERED

Attribute Usage Matrix:

	A1 Course id	A2 Grade	A3 Semester	A4 Roll number
Q1	0	0	0	0
Q2	0	0	0	0
Q3	0	1	0	1
Q4	0	0	0	0
Q5	0	0	0	1
Q6	0	0	0	0
Q7	0	0	0	1
Q8	0	0	0	0

Attribute Affinity Matrix:

	A1	A2	A3	A4
A1	0	0	0	0
A2	0	30	0	30
A3	0	0	0	0
A4	0	30	0	150

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 STUDENT AND HOSTELROOM relations.

8. 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:

Roll number(10)	Student Name(10)	DOB(3)	Year Enrolled(4)	Program(10)	Specialization(10)	CGPA(2)	Placement id(2)

Fragment-2:

Room number(10)	Student Name(10)	Mess id(2)	Hostelname(2)	Fine(2)	Rollnumber(2)

Fragment-3:

Placement id(2)	Student name(10)	CTC(2)	Recruitment type(10)	Company name (10)

Fragment-4:

Course id(10)	Grade(2)	Semester(10)	Roll number(10)

Fragment-5:

Book id (2)	Book name (10)	Date of barrow(4)	Status(10)	Fine(10)	Roll number(2)

Fragment-6:

Course id (10)	Course name (10)	Credits (2)

Fragment-7:

Student name (10)	Academic fee (2)	Fee receipt number (10)	Roll number (10)

Fragment-8:

Company id (2)	Company name (10)	Company type (10)	Phone number (10)	Address (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 8 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	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	49
F8	800	42	33600	34

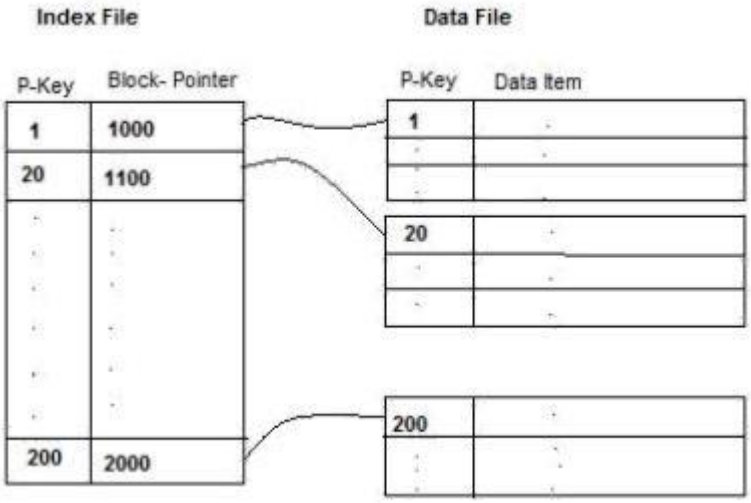
A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure. Indexes are used to quickly locate data without having to search every row in a database table every time 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 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 F4, in the worst case, all of its 57 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 = 8B

Block Size = 1024B

Fragment	**No. of Blocks**	**Size of the index table (B)**	**Total Size(B)**	**No. of index Blocks**
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
F8	34	10+8=18	612	11

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.

9. 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 = 1856 bytes

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 disk)

10. 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 * 10^6 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
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 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	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
F8	7201	7501	6402	6702

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
F8	S1	-	-	0	0
	S2	-	-	0	0
	S3	-	-	0	0
	S4	Q4	1*50*300	15000	15000

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	F1,F2,F3,F4,F5,F7,F8

11. 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 KB